NPS-59CI72121A

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THREE DIMENSIONAL FINITE ELEMENT STUDIES

PART ONE: SERVICE ROUTINES

by

Gilles Cantin

December 1972

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral Mason Freeman
Superintendent

M. U. Clauser
Provost

ABSTRACT:

In this report, service routines are developed for linear, quadratic and cubic finite elements for two and three dimensional regions. Also, an equation solver of very large capacity is developed for systems of linear equations where the matrix of coefficients is symmetrical, positive definite and tightly banded along the main diagonal.

This task was supported by: Naval Weapons Center, China Lake, California, Work Request No. 1-0029

Table of Contents

INTRODUCTION

The stress-strain analysis of solid propellant rocket motors has been the object of numerous investigations in recent years. Various approximate methods of analysis have been proposed. The Finite Element method, without a doubt, showed the greatest promise of providing the information needed in the evaluation of real motors. In principle the method can provide analyses of any three dimensional bodies. Any number of material and geometrical discontinuities can be admitted. The method also allows for almost any type of static, dynamic and thermal loadings to be applied. In practice, however, this has proved to be a formidable computational task even for the largest digital computers in existence.

An elegant compromise was obtained with the development of axisymmetric ring elements (1)*. However this compromise eliminated the possibility of solving for real motors with star shaped grains. In those cases it has generally been assumed that a plane strain analysis using elements of triangular shape would provide enough information for design. The constant strain finite element triangle has been used ad nauseam in these studies, in spite of the fact that many other elements have been shown to provide much better results for the same computational effort (2). When the perforation changes shape, the combination of ring

*Numbers in parenthesis refer to the reference list at the end.

elements and constant strain triangle cannot give reliable results.

Recently the introduction of isoparametric elements in the literature has given a new impetus to attempts at solving real three-dimensional motors (3), (4), (5). It is to such a task that we address ourselves. In this work we shall not attempt to account for real viscoelastic behavior, what we shall present is a linear elastic analysis method for three-dimensional objects. We will show that isoparametric elements can be economically generated in the computer, even for the monstrous (96 x 96), 32 nodal point bricks. A massive system of linear equations with a large half-bandwidth normally results from the analysis of realistic problems. For these systems of equations we propose a new solver (6).

Chapter I. Isoparametric Elements

In this chapter we describe isoparametric elements and the numerical techniques used to construct them.

1.1 Advantages of isoparametric elements

The development of isoparametric elements marked an important breakthrough in the finite element method (4). Until then, a great variety of elements were developed to solve specific types of problems. The most successful elements satisfied all the requirements to insure convergence to an exact result by mesh refinement. However the solution of many problems required a combination of various elements which could not satisfy all the required conditions for successful convergence, when assembled to solve one problem. An example of such a situation is: an axisymmetric system where truncated conical ring elements are used to model a case around a rocket motor and ring elements of triangular cross section are used to model the grain (1). The usual conical element admits transverse displacements given by a cubic polynomial where as the grain ring elements will allow only linear displacement in the same direction. Obviously, adjacent ring and case elements cannot maintain continuity of displacements under strain.

Isoparametric element families on the other hand can be developed to yield compatible elements of any type: line, surface or volume. Furthermore these elements admit curved boundaries, resulting in accurate modeling of complex shapes even with very few elements. The

results obtained in references (3), (4) and (5) show that good results are obtained with very coarse mesh.

1.2 Stiffness matrix of a finite element

The strain energy of an element having a stiffness matrix [K] can be written as:

$$U_s = \frac{1}{2} \langle \tilde{u}_i \rangle \, [K] \, \{\tilde{u}_i\} \tag{1}$$

where the vector $\{\tilde{u}_i\}$ represents all the nodal coordinates of the element.

The same energy can also be written from fundamental principles as follows:

$$U_s = \frac{1}{2} \int_v \langle \sigma_i \rangle \, \{\epsilon_i\} \, dv \tag{2}$$

where $\{\sigma_i\}$ is a vector of all the stress components and $\{\epsilon_i\}$ a vector of corresponding strain components in the volume V. The constitutive laws can be written in matrix form as follows:

$$\{\sigma_i\} = [\mathcal{E}] \{\epsilon_i\} \tag{3}$$

where $[\mathcal{E}]$ is a symmetric matrix of material properties. The strain displacement relations are also written in matrix form as follows:

$$\{\epsilon_i\} = [\varphi] \begin{Bmatrix} u(x, y, z) \\ v(x, y, z) \\ w(x, y, z) \end{Bmatrix} \tag{4}$$

where matrix $[\varphi]$ is in general a rectangular matrix of differential operators, and the vector $\langle u(x, y, z), v(x, y, z), w(x, y, z) \rangle^T$ represents

the three components of displacements in an appropriate global system of reference, (x, y, z). Finally the displacement functions are assumed to be of the following form:

$$\begin{Bmatrix} u(x, y, z) \\ v(x, y, z) \\ w(x, y, z) \end{Bmatrix} = [h] \{\tilde{u}_i\} \tag{5}$$

where $[h]$ is a rectangular matrix of carefully selected functions of (x, y, z). These functions must be selected to insure compatibility across common boundaries, to insure that rigid body motions will result in strainless states and finally to admit at least constant straining states.

After substitution of (5), (4) and (3) into (2) and using the definition:

$$[h^*] = [\varphi] [h] \tag{6}$$

we obtain the following result:

$$U_s = \frac{1}{2} \langle \tilde{u}_i \rangle \left(\int_v [h^*]^T [e] [h^*] \, dv \right) \{\tilde{u}_i\} \tag{7}$$

The stiffness matrix is then:

$$[K] = \int_v [h^*]^T [e] [h^*] \, dv \tag{8}$$

1.3 The isoparametric concept

In isoparametric elements, the displacements are obtained in terms of shape functions defined in a system of curvilinear coordinates particular to one element.

$$u(x, y, z) = N_i(\xi,\eta,\zeta)u_i$$
$$v(x, y, z) = N_i(\xi,\eta,\zeta)v_i \qquad (9)$$
$$w(x, y, z) = N_i(\xi,\eta,\zeta)w_i$$

where $N_i(\xi,\eta,\zeta)$ are the shape functions and u_i, v_i, w_i are nodal values of the displacement components in the global system (x, y, z). The global system of reference is then related to the curvilinear coordinates by the same shape functions as follows:

$$x(\xi,\eta,\zeta) = N_i(\xi,\eta,\zeta)x_i$$
$$y(\xi,\eta,\zeta) = N_i(\xi,\eta,\zeta)y_i \qquad (10)$$
$$z(\xi,\eta,\zeta) = N_i(\xi,\eta,\zeta)z_i$$

where x_i, y_i and z_i are the nodal coordinates of the element in the global system (x, y, z). It is not our intention here to trace the history of such a concept. The readers are referred to references (3) and (4) for the details and justification of the concept. The remaining task is to construct the shape functions. We have taken these functions from reference (4). They will be found in section 1.6, 1.7 and 1.8.

1.4 The Jacobian Matrix

In equation (6), the derivatives of the shape functions in terms of the coordinates of the global system (x, y, z) are needed. These operations require the Jacobian of the transformation:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \cdot \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} \tag{11}$$

where the square matrix is the Jacobian [J]. Using equation (10) the Jacobian matrix is easily evaluated as follows:

$$[J] = \begin{bmatrix} \frac{\partial N_1}{\partial \xi}, & \frac{\partial N_2}{\partial \xi}, & \cdots \\ \frac{\partial N_1}{\partial \eta}, & \frac{\partial N_2}{\partial \eta}, & \cdots \\ \frac{\partial N_1}{\partial \zeta}, & \frac{\partial N_2}{\partial \zeta}, & \cdots \end{bmatrix} \cdot \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ . & . & . \\ . & . & . \\ . & . & . \\ . & . & . \end{bmatrix} \tag{12}$$

When the Jacobian is known the required derivatives are easily obtained. The element of volume is then transformed as follows:

$$dV = \det(J)\, d\xi\, d\eta\, d\zeta \tag{13}$$

At this time everything is ideally suited for Gaussian numerical integration. This will be explained later.

1.5 Shape functions

As mentioned previously we take the displacement functions directly from reference (4). We write these shape functions for three dimensional

elements since they automatically contain the information for two and one dimensional elements. We also restrict our attention to tri-linear, tri-quadratic and tri-cubic polynomial expressions

1.6 Linear elements

The shape functions are:

$$N_i = \frac{1}{8}(1 + \xi_o)(1 + \eta_o)(1 + \zeta_o) \quad i = 1, 2, \ldots, 8 \qquad (14)$$

where $\xi_o = \xi\xi_i$ and ξ_i is ± 1, similarly for η_o and ζ_o.

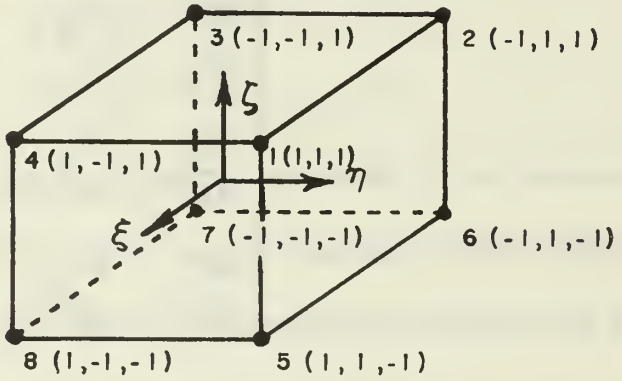


Figure 1. Linear element (8 nodal points)

1.7 Quadratic elements

See figure 2 for the identification of the nodal points.

Corner nodes: 1, 3, 5, 7, 13, 15, 17 and 19

$$N_i = \frac{1}{8}(1 + \xi_o)(1 + \eta_o)(1 + \zeta_o)(\xi_o + \eta_o + \zeta_o - 2) \qquad (15)$$

where $\xi_o = \xi\xi_i$ and ξ_i is ± 1, similarly for η_o and ζ_o.

Mid side nodes: 2, 6, 18 and 14

$$N_i = \frac{1}{4}(1 - \xi^2)(1 + \eta_o)(1 + \zeta_o) \qquad (16)$$

Mid side nodes: 4, 8, 20 and 16

$$N_i = \frac{1}{4}(1 - \eta^2)(1 + \zeta_o)(1 + \xi_o) \tag{17}$$

Mid side nodes: 9, 10, 11 and 12

$$N_i = \frac{1}{4}(1 - \zeta^2)(1 + \xi_o)(1 + \eta_o) \tag{18}$$

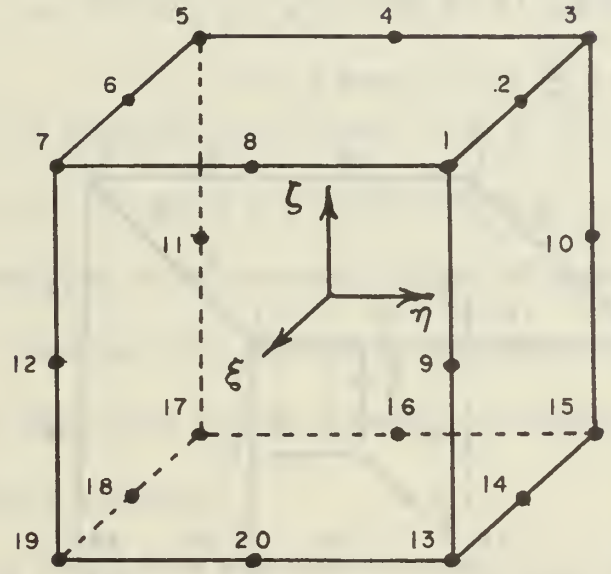


Figure 2. Quadratic element (20 nodal points)

1.8 Cubic elements

See figure 3 for the identification of the nodal points.

Corner nodes 1, 4, 7, 10, 21, 24, 27 and 30

$$N_i = \frac{1}{64}(1 + \xi_o)(1 + \eta_o)(1 + \zeta_o)[9(\xi^2 + \eta^2 + \zeta^2) - 19] \tag{19}$$

Mid side nodes: 2, 9, 29, 22, 3, 8, 28 and 23

$$N_i = \frac{9}{64}(1 - \xi^2)(1 + 9\xi_o)(1 + \eta_o)(1 + \zeta_o) \tag{20}$$

where $\xi_i = \pm \frac{1}{3}$, $\eta_i = \pm 1$ and $\zeta_i = \pm 1$

Mid side nodes: 12, 32, 25, 5, 11, 31, 26 and 6

$$N_i = \frac{9}{64}(1 - \eta^2)(1 + 9\eta_o)(1 + \zeta_o)(1 + \xi_o) \qquad (21)$$

where $\eta_i = \pm \frac{1}{3}$, $\xi_i = \pm 1$ and $\zeta_i = \pm 1$

Mid side nodes: 13, 14, 15, 16, 17, 18, 19 and 20

$$N_i = \frac{9}{64}(1 = \zeta^2)(1 + 9\zeta_o)(1 + \xi_o)(1 + \eta_o) \qquad (22)$$

where $\zeta_i = \pm \frac{1}{3}$, $\xi_i = \pm 1$ and $\eta_i = \pm 1$.

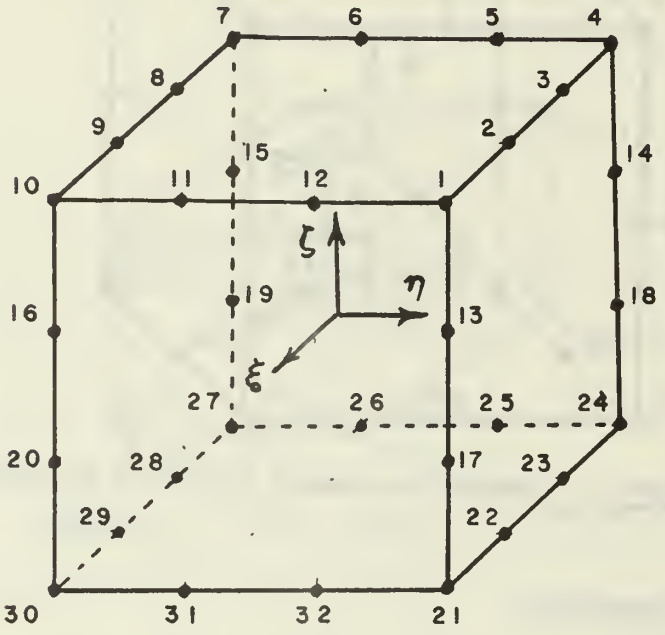


Figure 3. Cubic element (32 nodal points)

In all of these formulas if one coordinate , for example $\zeta = \zeta_o = 1$, the formulas degenerate to the corresponding correct formulas for the two dimensional element in plane (ξ, η). The same is true for line elements.

1.9 Numerical integration

When all the information contained in sections 1.4 to 1.8 is substituted into equation (8) we finally obtain an expression of the following form:

$$[K] = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} [\, k(\xi, \eta, \zeta)] \, d\xi \, d\eta \, d\zeta \qquad (23)$$

In this last expression the matrix $k(\xi, \eta, \zeta)$ is composed of elements in which the inverse of the Jacobian matrix is involved, furthermore, all the elements must be multiplied by the determinant of the Jacobian matrix. Obviously the process of reduction to the form shown in equation (23) is not trivial, however, a computational algorithm is not too difficult to obtain.

Once the form of equation (23) is obtained, Gaussian integration is straightforward. The number of Gauss points to be used in this integration is not easy to determine, especially if the element has irregular boundaries. For trirectangular elements, two, four and five Gauss ordinates in each direction of the local system of reference will give exact results for the linear, quadratic and cubic elements respectively. For elements having curved boundaries the same number of Gauss ordinates can only give approximate values to the stiffness matrix. We experimented with various numbers of Gauss points in those cases where the results were approximate, we observed that the dominant eigenvalues of the stiffness matrices were not sensitive

to the number of Gauss ordinates. We also observed that the trace of an approximate stiffness matrix was always smaller than the trace of the exact matrix. By increasing the number of Gauss points, the trace of the approximate matrix always approached the exact value from the low side. This error may actually be beneficial, since the trace of a stiffness matrix is known to be always greater than the trace of the actual stiffness of a real structure. For this reason we have adopted to use 2, 4 and 5 Gauss ordinates in the generation of the stiffness matrices. It will be necessary to show with actual problems whether or not this compromise is valid in general.

Chapter II. Computer Programs for the Generation of Stiffness Matrices

In this chapter we describe several computer programs used to evaluate the stiffness matrices of linear, quadratic and cubic elements for two and three dimensional bodies.

2.1 Stiffness matrices for two dimensional bodies

Appendix one contains the listing of a computer program that was used to evaluate the stiffness matrices of two-dimensional elements. Although the listing contains only one quadrature formula with five Gauss ordinates, an exhaustive series of tests was conducted with formulas using from two to ten Gauss ordinates. The five point formula was finally adopted for practical considerations; the formation times for one element were: 0.3, 1.0, 2.5 seconds for linear, quadratic and cubic elements respectively, and this was judged to be adequate for this phase of our research.

Following the strategy developed in section 1.9, two subroutines are needed to do all the work. The first subroutine named QUAD 5 performs the numerical integration, the second named FORMK assembles the stiffness matrix $k(\xi, \eta, \zeta)$ where (ξ, η, ζ) take their values at one of the Gauss points. Upon exit from the subroutines, the user will obtain the stiffness matrix of one of the elements in this family and also the transformation matrices between nodal strains (stresses) and nodal displacements. The calling statement is given in the appendix.

2.2 Stiffness matrices for three dimensional bodies

Appendix two contains three separate programs for the generation of the stiffnesses of linear, quadratic and cubic elements for three dimensional bodies. The three programs are similar in structure to the above program, they were coded separately for convenience during the testing phase of this study. The FORTRAN calling statements are all contained in the appendix. The number of Gauss point used was determined in section 1.9 for these elements.

Comments distributed throughout these programs should make the codes intelligible to an experienced user of finite element codes.

2.3 Numerical testing of stiffness matrices

The subroutines mentioned in the previous two articles were tested under the control of another program not listed here. Eigenvalues and eigenvectors were calculated for all these elements. Two dimensional elements always had three zero eigenvalues and three dimensional elements always had six zero eigenvalues. Static equilibrium checks were also applied to all elements and equilibrium was always satisfied.

2.4 Hybrid elements

A remarkable property of isoparametric elements is that once a high order element is obtained it can easily be degenerated to a lower order by simple matrix transformations. For example let us consider the eight nodal point two-dimensional element:

$$\{F\} = [K]\{\tilde{u}_i\} \tag{24}$$

where $[K]$ is the stiffness matrix and $\{\tilde{u}_i\}$ is the vector of nodal displacements;

$$\{\tilde{u}_i\} = \langle u_1, v_1, u_2, v_2, u_3, v_3, u_4, u_4, u_5, v_5, u_6, v_6, u_7, v_7, u_8, v_8 \rangle^T \tag{25}$$

see figure 4.

To degenerate this element to the corresponding four nodal point element, it is only necessary to impose a simple constraint to the mid points of the element. If the mid-points are such that their nodal displacements are equal to the average of the displacements of their corresponding corner points the following transformation matrix is readily constructed:

$$\{\tilde{u}_i\} = [T]\{\tilde{u}_i^*\} \tag{26}$$

where

$$\{\tilde{u}_i^*\} = \langle u_1, v_1, u_3, v_3, u_5, v_5, u_7, v_7 \rangle^T \tag{27}$$

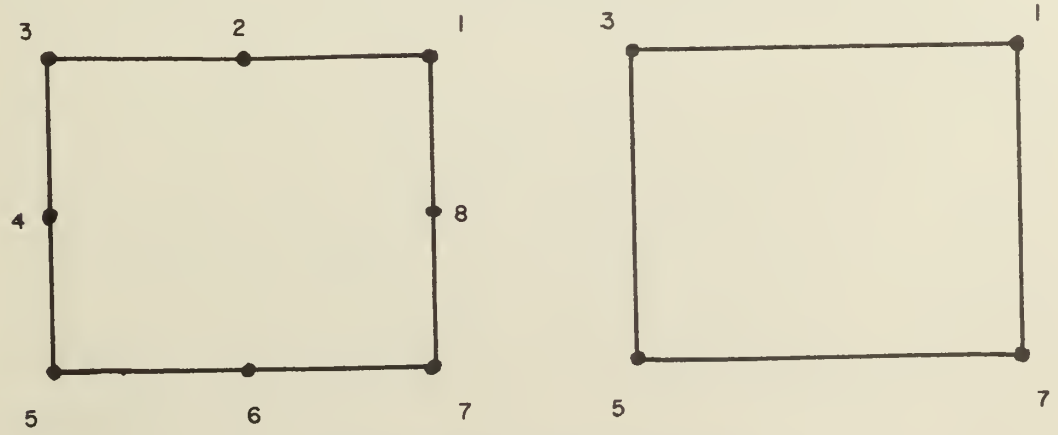


Figure 4. Hybrid elements

The matrix [T] in this case is simply:

$$[T] = \begin{bmatrix}
1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.5 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\
0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 \\
0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5
\end{bmatrix} \qquad (28)$$

The stiffness matrix for the four nodal point element is simply:

$$[K^*] = [T]^T [K][T] \tag{29}$$

If a five nodal point element is desired, for example, suppose we would like to keep nodal point 8 in the above example, then two columns would be added to the matrix [T]. Column 9 of [T] would have a one in row 15 and column 10 a one in row 16, all the other terms of rows 15 and 16 would be zeros. The congruent transformation (29) would give the stiffness of the hybrid element.

With such elements it is possible to combine different types of elements in the solution of one problem and still preserve all the conditions guaranteeing convergence to an exact result by mesh size reduction.

Chapter III. An Equation Solver of Very Large Capacity

In this chapter a very simple equation solver of very large capacity is developed. The method used does not require a very large in-core memory; however, a fast random access device like a magnetic disk drive is required.

3.1 Introduction

The solution of very large systems of linear equations,

$$\{R\} = [K] \cdot \{r\} \tag{30}$$

where [K] is symmetric and positive definite, is an activity that must be entered by the stress analyst when using the finite element method. When the first automatic stress analysis programs were developed, the Gauss-Seidel iteration method, with various over relaxation schemes, was almost universally employed to solve the equations. It was soon realized that, because of the properties of [K], a direct Gaussian elimination algorithm would be much more economical of computer time. However, matrix [K], must be in core during elimination, and this puts a very stringent limitation on the total number of equations that can be solved.

For the purpose of comparing various methods, we will presume that we have at our disposition a machine with 32,000 words of core storage. Of the total memory, let us assume that approximately 25,000 words are available for the storage of [K]. For such a machine, then, approximately 160 equations, at most, can be solved by this

method.

Because the matrix [K] is usually sparse, a special ordering of the unknowns can produce a matrix [K] that is tightly banded about the main diagonal. Taking advantage of this character of [K], several programs were devised that required only (N.M) words for the storage of [K]. (See Figure 5 for the meaning of N and M). The short program listed in Appendix 3 is of that character. If M is very small, then the number of equations that can be solved can become fairly large. For example, using the same machine as above, if M = 50, then N = 500.

With the development of two and three dimensional elements, however, this improvement proved insufficient. These problems frequently lead to systems of several thousands of equations with a half band width in the hundreds. Equation solvers were then developed that did not require to have all the equations in core. Professor E. Wilson of the University of California at Berkeley (U. S.) several years ago developed a band solver that could be used with tape storage to solve large systems. The method used required (2M.M) words of in-core storage for [K] and the total number of equations that could be solved was only limited by the capacity of a tape. For the machine used above, the maximum value of M is approximately 110. Subsequently, as mentioned in Reference (8), several workers developed algorithms requiring only (M + 2)(M + 3)/2 words of core storage. For the machine

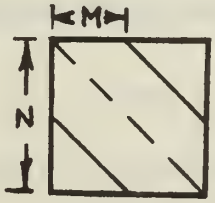
used above, the maximum value of M is approximately 220.

With large systems of equations, another difficulty becomes evident. Round off starts to play a very important role. An obvious remedy is to convert everything to double precision. However, this roughly doubles the core storage requirement. Round off can sometimes be eliminated by using the same programs to calculate corrections to the answer by an iterative technique. However, such corrections are not always valid when the matrix [K] is locally ill-conditioned. For this reason, many workers prefer to use double precision all the way. The block solver presented here uses double precision everywhere.

As mentioned in References (9) and (10), no method has been found that is faster and more accurate than Gaussian elimination for a system in which the matrix of coefficients is dense. For dense system banded along the main diagonal, it is reasonable to conjecture that the same conclusion will hold if the algorithm is modified to eliminate all operations involving the off diagonal zeros. Improvements can only be accomplished by taking advantage of the sparsity within the band, as shown in References (11) and (12). These improvements, however, are obtained at the cost of a more elaborate code to recognize the particular type of sparsity that can lead to important reductions of computations. The economies of storage, on the other hand, are not sufficient to allow for the solution of very large systems.

Frontal solvers, as described in References (13) and (8), also produce savings of storage and execution time, but again prohibit the solution of very large systems.

Our primary concern here was to design a solver that would permit the solution of very large systems even with a small computer. The block solver proposed in this paper does not pose any limits on the half band width nor on the total number of equations, as long as storage is available on a fast random access external device. The program contained in Appendix 4 requires 5800 bytes* of storage for the object codes and a working space depending on the block size selected; for block sizes of (25, 50, 100) the storage requirements are (15600, 61200, 242400) bytes respectively. As mentioned above, a fast random access facility is required. In our computer center, we used IMB/2314 disk packs. On such a device, 55000 records, 400 bytes long, can be stored. This is enough for 2,750,000 double words. For example, 5000 equations with a half band width of 451 would fit with a complete load vector on one disk.



M is the half band width

N is the total number of equations

Figure 5. Banded Character of K

*A byte is a word segment of 8 bits of information.

3.2 Elimination by blocks

Consider the system of equations shown below:

$$\begin{Bmatrix} R_1 \\ R_2 \\ R_3 \\ \cdot \\ R_M \\ \cdot \\ R_I \\ \cdot \\ R_N \end{Bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} & \cdot & \cdot & K_{1M} & \cdot & \cdot & \cdot \\ K_{12}^T & K_{22} & K_{23} & K_{24} & \cdot & \cdot & K_{2,M+1} & \cdot & \cdot \\ K_{13}^T & K_{23}^T & K_{33} & K_{34} & K_{35} & \cdot & \cdot & K_{3,M+2} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ K_{1M}^T & K_{2M}^T & \cdot & \cdot & K_{M-1,M}^T & K_{MM} & K_{M,M+1} & \cdot & K_{M,2M} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & K_{1-M+1,I}^T & \cdots & K_{I-1,I}^T & K_{II} & K_{I,I+1} & \cdot & \cdots & K_{I,I+M-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & K_{N-M+1,N}^T & \cdot & \cdot & \cdot & K_{N-1,N}^T & K_{NN} \end{bmatrix} \cdot \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \cdot \\ r_M \\ \cdot \\ r_1 \\ \cdot \\ r_n \end{Bmatrix} \tag{31}$$

The first equation is solved for r_1 to get:

$$r_1 = K_{11}^{-1} (R_1 - K_{12}\, r_2 - K_{13}\, r_3 - \ldots - K_{1M}\, r_M) \tag{32}$$

After substitution in equations 2 to M, we obtain the following results:

$$\bar{R}_2 = R_2 - K_{12}^T\, K_{11}^{-1}\, R_1 \tag{33}$$

$$\bar{R}_M = R_M - K_{1M}^T\, K_{11}^{-1}\, R_1 \tag{34}$$

for the load vectors. All the coefficients in rows 2 to M from Column 2 to M are reduced by similar operations; a typical example for term K_{34} is:

$$\bar{K}_{34} = K_{34} - K_{13}^{T} K_{11}^{-1} K_{14} \tag{35}$$

Then, Equation 2 is solved for r_2 and substituted in Equation 3 to M + 1 and the entire process is repeated until we get only the last equation:

$$\bar{R}_N = \bar{K}_{NN} r_N \tag{36}$$

This last equation can now be solved for r_N and a simple process of back substitution in reverse order gives all the unknowns.

The elimination process described above preserves the symmetry of [K]. This process is valid, whether the elements of [K] are individual coefficients or square submatrices of coefficients. From now on, we presume that the $[K_{IJ}]$ are submatrices that we will call blocks. A careful examination of the elimination steps reveals that no more than three different blocks of coefficients are needed. With this in mind, the rest of the task is trivial. The simple solver contained in Appendix 3 was modified to do the elimination by blocks. Appendix 4 contains the listings of the four subroutines needed.

3.3 Listings

Subroutine SOLVE is self-explanatory and needs no elaboration, except as to the manner in which the equations must be stored. The blocks are stored sequentially by rows, as shown below. At the end of each row, a block of load vectors is present:

$$
\begin{array}{ll}
K_{11}\ K_{12}\ K_{13}\ K_{14} \cdots \cdots K_{1m} & R_1 \\
K_{22}\ K_{23}\ K_{24}\ K_{25} \cdots \cdots K_{2m} & R_2 \\
\cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots & \cdots \\
K_{II}\ K_{I,I+1}\ \cdots\ \cdots\ \cdots\ \cdots & R_I \\
\cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots & \cdots \\
K_{n-1,n-1}\ K_{n-1,n}\ \cdots\ \cdots\ \cdots & R_{n-1} \\
K_{nn} \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots & R_n
\end{array}
\qquad (37)
$$

The symbols m and n are the number of blocks per row and column, respectively.

This method of storage leaves a few blocks of unused storage. However, the last row of blocks is needed for temporary storage during the elimination. Three service subroutines are needed in SOLVE. The multiplication and inversion operations are standard and need no words of explanation. The subroutine WRDISK/RDDISK, however, is hardware dependent and uses FORTRAN commands particular to IBM/360 installations (Reference 14). This feature, although undesirable, is not a very serious drawback because every installation using fast random access devices has specialized commands to use these facilities. For other installations, this subroutine would have to be rewritten to conform with the available hardware. The listings are self-explanatory.

3.4 Timing

The use of the program listed in Appendix 4 poses a serious practical problem. The solution of large systems of equations requires a large amount of computer time. The most important factors affecting the required time in the system proposed are: the number of calls to the WRDISK/RDDISK routine, and the size of the blocks (N_s). In the program, the number of calls to WRDISK/RDDISK has been reduced as much as we could. In actual tests, the object codes were optimized, with respect to execution time, by the compilers that produced these object codes. We used the H compiler of release 18 of the IBM/360 operating system. A system of 500 equations with a half band width of 151 was solved with three different block sizes, (25, 50, 100). The execution times were (510, 402, 297) seconds, respectively. On the basis of this test, it seems that the block-size should be as large as the computer will allow, in order to reduce the execution time. For a block-size of 50, the following empirical formula gives an estimate of the required time:

$$T = 1.857\ nm^2 + 0.102\ n^2m - 0.322\ n^2 + 4.682\ nm - 5.230\ m^2 \quad (38)$$

The formula was obtained with the tests shown in Table 1. For the first test n = 4, m = 3, it was possible to obtain an in-core solution with the subroutine shown in Appendix 3. The execution time was 61.3 seconds for the in-core solution, as compared to 75.7 seconds for the block solver. This is indeed a very small penalty if we also compare the

storage requirements of the two solvers; 196K bytes for the in-core solver and 122K bytes for the block solver.

In a recent study, Reference (15), various equation solvers were compared. Using a sliding block technique similar to that mentioned in Reference (8), a system of 1080 equations with a half-band width of 357 was solved in 1365.1 seconds. The machine used was a CDC 6600 and the method used required 73728 words of core storage. Test Number 5 in Table 1 shows that 1000 equations with a half-band width of 351 required 2989.8 seconds and used 15250 double words of core storage. Again, this is a comparison that seems very favorable to the block solver, if we keep in mind that the CDC6600 is a much faster machine than the IBM/360/67. Average execution times for division, multiplication and addition are: (2.9, 1.0, 0.4) micro-seconds for the CDC machine and (13.9, 10.6, 8.0) for the IBM machine.

3.5 Accuracy

The most critical operation, with respect to round-off, is the inversion of the diagonal blocks. If the block is singular, the program will print a diagnostic to this effect. However, this is not likely to happen very often. A much more serious difficulty comes from blocks that are nearly singular and this must also be detected. For this reason, the inversion routine will calculate an estimate of the round-off value of zero as the trace of the block multiplied by 10^{-12} and report that the block is nearly singular, if any diagonal term becomes smaller

TABLE 1 TIMING TESTS*

n (Number of blocks per column)	m (Number of blocks per row)	N (Number of equations)	M (Half-band width)	T (Time in seconds)
4	3	200	101	75.7
100	3	5000	101	2977.4
10	5	500	201	586.5
50	5	2500	201	3834.8
20	8	1000	351	2989.8

*The blocksize used was 50 and the total storage requirement of the test program was 122K bytes or 15250 double words.

than this value during inversion. Besides these diagnostics, the program will also print a condition number for each diagonal block. This condition number is obtained as the product of the maximum column sum of the block before and after inversion, following Wilkinson's perturbation theory. With this condition number, an estimate of the relative error in the answer can be obtained from:

$$E(\%) \leq c \cdot \frac{\|P\|}{\|R\|} \cdot 100 \qquad (39)$$

where c is the maximum value of all condition numbers of the system, $\|P\|$ is the sum of the absolute values of the estimated errors in the loads and $\|R\|$ is the sum of the absolute values of the loads.

3.6 Modifications

The program is presented in its simplest form; no facilities for iteration and multiple-load vectors are provided for. Users with particular needs should have no difficulty modifying the program listed, in order to take advantage of hardware available in their computing facilities.

3.7 Conclusion

The program presented here provides a very simple answer to a most difficult practical difficulty. When a problem is too big for a band or a frontal solver, the block solver will still give a solution, even with a very small computer, provided one is ready to pay a penalty in execution time.

Chapter IV. Conclusions

The publication of this report was delayed in order to verify the integrity of the service routines that it contains. This has now been completed and it can be reported that the routines perform as expected.

4.1 Stress-strain analysis codes

Several computer codes for mesh generation and consistent load vectors computation, as well as two basic stress strain programs for two and three dimensional problems, have been completed and tested (16), (17) and (18). From that work it may be concluded that these elements will shed significant new light on the stress analysis of rocket motors.

4.2 Reduced Gaussian Integration

In both reference (17) and (18), it is reported that the Gaussian integration mentioned in article 1.9 leads to improved results when only 2 and 3 Gauss ordinates are uses, respectively, in the generation of the 20 and 32 nodal point elements. This reduced integration also leads to significant economies in the total solution times.

4.3 Surface stresses

In some problems it was observed that the stress and strain values obtained at the nodes were in error on the external surfaces of the objects analysed. Internal nodes, however, gave the best results. This matter deserves further investigations.

4.4 Future work

The programs developed during the course of this investigation will be refined and released as soon as they are ready. In the meantime, potential users can refer to reference (17) for a first release of the three-dimensional programs.

Appendix I. Code for the generation of two-dimensional elements

```
      SUBROUTINE QUAD5(STK,AK,B,SS,SN,NS,N)                              QFV00010
C
C     QUAD5 IS A NUMERICAL INTEGRATION SUBROUTINE THAT FORMS THE STIFFNESS
C     MATRIX FOR ANY OF THE THREE QUADRILATERAL ELEMENTS IN THIS FAMILY.
C     GAUSSIAN QUADRATURE WITH FIVE ORDINATES IS USED THROUGHOUT.
C
C     THE SUBROUTINE IS CALLED WITH THE FOLLOWING STATEMENT:
C     CALL QUAD(STK,AK,SS,SN,NS,N)
C     THE CALLING PROGRAM MUST CONTAIN THE FOLLOWING COMMON STATEMENT:
C     COMMON COORD(12,2),ELAST(3,3)
C          STK   IS AN ARRAY DIMENSIONED NXN THAT WILL CONTAIN THE STIFFNESS
C                MATRIX UPON RETURN TO THE CALLING PROGRAM
C          SS AND SN ARE THE STRESS AND STRAIN VERSUS NODAL POINT DISPLACEMENT
C                TRANSFORMATION MATRIX, BOTH ARE DIMENSIONED NSXN WHERE
C                NS IS 3*NPT AND N IS 2*NPT
C          AK    IS AN ARRAY DIMENSIONED NXN REQUIRED IN FORMK
C          B     IS AN ARRAY DIMENSIONED 2XN REQUIRED IN FORMK
C
C
C          COORD(12,2) IS AN ARRAY CONTAINING THE COORDINATES OF ALL
C                      THE NODAL POINTS OF THE ELEMENT TO BE CONSTRUCTED
C          ELAST(3,3)  IS AN ARRAY CONTAINING ALL THE ELASTIC CONSTANTS
C                      OF THE ELEMENT, THE ORDER OF THE STRESS COMPONENTS
C                      IS: SIGMAX,SIGMAY,TAUXY
C
C                ETA                      ETA                      ETA
C                 +                        +                        +
C          U2-----+------U1         U3-----U2-----U1         U4--U3--U2--U1
C           |     +       |          |     +       |          |     +     |
C           |     +       |          |     +       |          |     +     |
C           |     +       |          |     +       |          |     +     |
C           |     +       |          |     +       |          |     +     |
C           |     +       |          |     +       |          U5    +     U12
C           |     +       |          |     +       |          |     +     |
C           |     +++++++++XI        U4    ++++++U8++XI       |     +++++++++XI
C           |             |          |           |             |           |
C           |             |          |           |             U6          U11
C           |             |          |           |             |           |
C           |             |          |           |             |           |
C          U3-----------U4         U5-----U6-----U7         U7--U8--U9--U10
C
      IMPLICIT REAL*8(A-H,O-Z)                                           QFV00020
      DIMENSION AK(N,N),STK(N,N),B(3,N)                                  QFV00030
      DIMENSION XI(5),AI(5),AIA(5,5)                                     QFV00040
      DIMENSION SS(NS,N),SN(NS,N)                                        QFV00041
```

```
      DIMENSION XYL(4,2),XYQ(8,2),XYC(12,2)                                 QFV00042
      COMMON COORD(12,2),ELAST(3,3)                                         QFV00043
      DATA XI/0.9061798459386640,0.5384693101056831,0.0D0,-0.538469310105QFV00050
     156831,-0.9061798459386640/                                            QFV00060
      DATA AI/0.2369268850561891,0.4786286704993665,0.5688888888888889,0QFV00070
     1.4786286704993665,0.2369268850561891/                                 QFV00080
      DATA XYL/1.0D0,-1.0D0,-1.0D0,1.0D0,1.0D0,1.0D0,-1.0D0,-1.0D0/         QFV00081
      DATA XYQ/1.0D0,0.0D0,-1.0D0,-1.0D0,-1.0D0,0.0D0,1.0D0,1.0D0,1.0D0,QFV00082
     11.0D0,1.0D0,0.0D0,-1.0D0,-1.0D0,-1.0D0,0.0D0/                         QFV00083
      DATA XYC/1.0D0,0.3333333333333333,-0.3333333333333333,-1.0D0,         QFV00084
     1-1.0D0,-1.0D0,-1.0D0,-0.3333333333333333,0.3333333333333333,1.0D0,QFV00085
     21.0D0,1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,0.3333333333333333,               QFV00086
     3-0.3333333333333333,-1.0D0,-1.0D0,-1.0D0,-1.0D0,-0.3333333333333333QFV00087
     43,0.3333333333333333/                                                 QFV00088
      DO 100 I=1,N                                                          QFV00090
      DO 100 J=1,N                                                          QFV00100
  100 STK(I,J)=0.0D0                                                        QFV00110
      DO 150 I=1,5                                                          QFV00120
      DO 150 J=1,5                                                          QFV00130
  150 AIA(I,J)=AI(I)*AI(J)                                                  QFV00140
      DO 200 I=1,5                                                          QFV00150
      X=XI(I)                                                               QFV00160
      DO 200 J=1,5                                                          QFV00170
      Y=XI(J)                                                               QFV00180
      CALL FORMK(AK,X,Y,B,N)                                                QFV00190
      DO 200 K=1,N                                                          QFV00200
      DO 200 L=1,N                                                          QFV00210
  200 STK(K,L)=STK(K,L)+AIA(J,I)*AK(K,L)                                    QFV00220
      DO 300 I=1,N                                                          QFV00230
      DO 300 J=I,N                                                          QFV00240
      STK(I,J)=(STK(I,J)+STK(J,I))*0.5D0                                    QFV00250
  300 STK(J,I)=STK(I,J)                                                     QFV00260
      NPT=N/2                                                               QFV00270
      IGO=NPT/4                                                             QFV00280
      DO 1000 I=1,NPT                                                       QFV00290
      GO TO (400,500,600),IGO                                               QFV00300
  400 X=XYL(I,1)                                                            QFV00310
      Y=XYL(I,2)                                                            QFV00320
      GO TO 700                                                             QFV00330
  500 X=XYQ(I,1)                                                            QFV00340
      Y=XYQ(I,2)                                                            QFV00350
      GO TO 700                                                             QFV00360
  600 X=XYC(I,1)                                                            QFV00370
      Y=XYC(I,2)                                                            QFV00380
  700 CALL FORMK(AK,X,Y,B,N)                                                QFV00390
      DO 800 L=1,3                                                          QFV00400
      DO 800 M=1,N                                                          QFV00410
      AK(L,M)=0.0D0                                                         QFV00420
```

```
      DO 800 NN=1,3                                                    QFV00430
  800 AK(L,M)=AK(L,M)+ELAST(L,NN)*B(NN,M)                              QFV00440
      II=3*(I-1)                                                       QFV00450
      DO 900 J=1,3                                                     QFV00460
      IJ=II+J                                                          QFV00470
      DO 900 K=1,N                                                     QFV00480
      SS(IJ,K)=AK(J,K)                                                 QFV00490
  900 SN(IJ,K)=B(J,K)                                                  QFV00500
 1000 CONTINUE                                                         QFV00510
      RETURN                                                           QFV00520
      END                                                              QFV00530
```

```
      SUBROUTINE FORMK(AK,X,Y,B,N)                                      FMK00010
C
C
C THIS SUBROUTINE FORMS THE STIFFNESS AND THE B MATRIX AS FUNCTIONS OF X AND Y
C FOR THE THREE DIFFERENT QUADRILATERAL ELEMENTS IN THIS FAMILY
C
C
      IMPLICIT REAL*8(A-H,O-Z)                                          FMK00020
      DIMENSION AJ(2,2),AJIN(2,2),DNX(2,12),W1(2,12),B(3,N)             FMK00030
     1,B1(24,3),AK(N,N)                                                 FMK00040
      COMMON COORD(12,2),ELAST(3,3)                                     FMK00050
C INITIALISE
      DO 10 I=1,3                                                       FMK00060
      DO 10 J=1,N                                                       FMK00070
   10 B(I,J)=0.0D0                                                      FMK00080
      ZERO=0.0D0                                                        FMK00090
      ONE=1.0D0                                                         FMK00100
      TWO=2.0D0                                                         FMK00110
      FOUR=4.0D0                                                        FMK00120
      NPT=N/2                                                           FMK00130
C
C FORM (2XNPT) MATRIX OF DERIVATIVES OF  THE INTERPOLAT.  FUNCT. WRT XI AND ETA
C
      IGO=N/8                                                           FMK00140
      GO TO (20,30,40),IGO                                              FMK00150
C
CC    LINEAR FUNCTIONS
C
   20 W1(1,3)=-(ONE-Y)/FOUR                                             FMK00160
      W1(1,4)=-W1(1,3)                                                  FMK00170
      W1(1,1)= (ONE+Y)/FOUR                                             FMK00180
      W1(1,2)=-W1(1,1)                                                  FMK00190
      W1(2,3)=-(ONE-X)/FOUR                                             FMK00200
      W1(2,4)=-(ONE+X)/FOUR                                             FMK00210
      W1(2,1)=-W1(2,4)                                                  FMK00220
      W1(2,2)=-W1(2,3)                                                  FMK00230
      GO TO 50                                                          FMK00240
   30 CONTINUE                                                          FMK00250
C
CC    QUADRATIC FUNCTIONS
C
      TXPY=TWO*X+Y                                                      FMK00260
      TXMY=TWO*X-Y                                                      FMK00270
      TYPX=TWO*Y+X                                                      FMK00280
      TYMX=TWO*Y-X                                                      FMK00290
      OMY=ONE-Y                                                         FMK00300
```

```
      OPY=ONE+Y                                                          FMK00310
      OPX=ONE+X                                                          FMK00320
      OMX=ONE-X                                                          FMK00330
      W1(1,5)=OMY*TXPY/FOUR                                              FMK00340
      W1(1,6)=-OMY*X                                                     FMK00350
      W1(1,7)=OMY*TXMY/FOUR                                              FMK00360
      W1(1,8)=OPY*CMY/TWO                                                FMK00370
      W1(1,1)=OPY*TXPY/FOUR                                              FMK00380
      W1(1,2)=-OPY*X                                                     FMK00390
      W1(1,3)= OPY*TXMY/FOUR                                             FMK00400
      W1(1,4)=-OPY*OMY/TWO                                               FMK00410
      W1(2,5)=OMX*TYPX/FOUR                                              FMK00420
      W1(2,6)=-OPX*OMX/TWO                                               FMK00430
      W1(2,7)=OPX*TYMX/FOUR                                              FMK00440
      W1(2,8)=-Y*CPX                                                     FMK00450
      W1(2,1)=OPX*TYPX/FOUR                                              FMK00460
      W1(2,2)=OPX*OMX/TWO                                                FMK00470
      W1(2,3)=CMX*TYMX/FOUR                                              FMK00480
      W1(2,4)=-Y*CMX                                                     FMK00490
      GO TO 50                                                           FMK00500
   40 CONTINUE                                                           FMK00510
C
C     CUBIC FUNCTICNS
C
      OPX=ONE+X                                                          FMK00520
      OPY=ONE+Y                                                          FMK00530
      OMX=ONE-X                                                          FMK00540
      OMY=ONE-Y                                                          FMK00550
      TPTMNX=3.0D0+2.0D0*X-9.0D0*X*X                                     FMK00560
      TMTMNX=3.0D0-2.0D0*X-9.0D0*X*X                                     FMK00570
      TPTMNY=3.0D0+2.0D0*Y-9.0D0*Y*Y                                     FMK00580
      TMTMNY=3.0D0-2.0D0*Y-9.0D0*Y*Y                                     FMK00590
      TYPO=3.0D0*Y+ONE                                                   FMK00600
      TYMO=3.0D0*Y-ONE                                                   FMK00610
      TXPO=3.0D0*X+ONE                                                   FMK00620
      TXMO=3.0D0*X-ONE                                                   FMK00630
      TMTPEX=10.0D0-27.0D0*X*X+18.0D0*X-9.0D0*Y*Y                        FMK00640
      TMTMEX=10.0D0-27.0D0*X*X-18.0D0*X-9.0D0*Y*Y                        FMK00650
      TMNPEY= 10.0D0-9.0D0*X*X+18.0D0*Y-27.0D0*Y*Y                       FMK00660
      TMNMEY= 10.0D0-9.0D0*X*X-18.0D0*Y-27.0D0*Y*Y                       FMK00670
      TTT=32.0D0                                                         FMK00680
      TTN=32.0D0/9.0D0                                                   FMK00690
      W1(1,7)= OMY*TMTPEX/TTT                                            FMK00700
      W1(1,8)=-TPTMNX*OMY/TTN                                            FMK00740
      W1(1,9)=TMTMNX*OMY/TTN                                             FMK00750
      W1(1,10)=-CMY*TMTMEX/TTT                                           FMK0071
      W1(1,11)=-OPY*OMY*TYMO/TTN                                         FMK00760
      W1(1,12)=OPY*OMY*TYPO/TTN                                          FMK00770
```

```
      W1(1,1)=-OPY*TMTMEX/TTT                                          FMK00720
      W1(1,2)=TMTMNX*OPY/TTN                                           FMK00780
      W1(1,3)=-TPTMNX*OPY/TTN                                          FMK00790
      W1(1,4)  =OPY*TMTPEX/TTT                                         FMK00730
      W1(1,5)  =-OPY*OMY*TYPO/TTN                                      FMK00800
      W1(1,6)  = OPY*OMY*TYMO/TTN                                      FMK00810
      W1(2,7)  = CMX*TMNPEY/TTT                                        FMK00820
      W1(2,8)= OPX*OMX*TXMO/TTN                                        FMK00860
      W1(2,9)=-OPX*OMX*TXPO/TTN                                        FMK00870
      W1(2,10) = CPX*TMNPEY/TTT                                        FMK00830
      W1(2,11)=-TPTMNY*OPX/TTN                                         FMK00880
      W1(2,12)= TMTMNY*OPX/TTN                                         FMK00890
      W1(2,1)  =-OPX*TMNMEY/TTT                                        FMK00840
      W1(2,2)=OPX*OMX*TXPO/TTN                                         FMK00900
      W1(2,3)=-OPX*OMX*TXMO/TTN                                        FMK00910
      W1(2,4)  =-OMX*TMNMEY/TTT                                        FMK00850
      W1(2,5)  =OMX*TMTMNY/TTN                                         FMK00920
      W1(2,6)  =-CMX*TPTMNY/TTN                                        FMK00930
   50 CONTINUE                                                         FMK00940
C
CC FORM JACOBIAN OF THE TRANSFORMATION IN AJ(I,J)
C
      DO 100 I=1,2                                                     FMK00950
      DO 100 J=1,2                                                     FMK00960
      AJ(I,J)=ZERO                                                     FMK00970
      DO 100 K=1,NPT                                                   FMK00980
  100 AJ(I,J)=AJ(I,J)+W1(I,K)*COORD(K,J)                               FMK00990
C
CC CALCULATE DETERMINANT OF THE JACOBIAN
C
      DTJ=AJ(1,1)*AJ(2,2)-AJ(1,2)*AJ(2,1)                              FMK01000
C
CC INVERT JACOBIAN IN AJIN
C
      AJIN(1,1)=AJ(2,2)/DTJ                                            FMK01010
      AJIN(1,2)=-AJ(1,2)/DTJ                                           FMK01020
      AJIN(2,1)=-AJ(2,1)/DTJ                                           FMK01030
      AJIN(2,2)=AJ(1,1)/DTJ                                            FMK01040
C
CC FORM (2XNPT) MATRIX OF DERIVATIVES OF INTERP. FUNCT WRT X AND Y
C
      DO 200 I=1,2                                                     FMK01050
      DO 200 J=1,NPT                                                   FMK01060
      DNX(I,J)=ZERO                                                    FMK01070
      DO 200 K=1,2                                                     FMK01080
  200 DNX(I,J)=DNX(I,J)+AJIN(I,K)*W1(K,J)                              FMK01090
C
C FORM B(I,J) MATRIX
```

```
C
      DO 300 I=1,NPT                                                    FMK01100
      IOD=2*I-1                                                         FMK01110
      IEV=2*I                                                           FMK01120
      B(1,IOD)=DNX(1,I)                                                 FMK01130
      B(2,IEV)=DNX(2,I)                                                 FMK01140
      B(3,IEV)=DNX(1,I)                                                 FMK01150
  300 B(3,IOD)=DNX(2,I)                                                 FMK01160
C
C  FORM STIFFNESS BY THE CONGRUENT TRANSFORMATION  BT*D*B
C
      DO 500 I=1,N                                                      FMK01170
      DO 500 J=1,3                                                      FMK01180
      B1(I,J)=ZERO                                                      FMK01190
      DO 500 K=1,3                                                      FMK01200
  500 B1(I,J)=B1(I,J)+B (K,I)*ELAST(K,J)                                FMK01210
      DO 600 I=1,N                                                      FMK01220
      DO 600 J=1,N                                                      FMK01230
      AK(I,J)=ZERO                                                      FMK01240
      DO 600 K=1,3                                                      FMK01250
  600 AK(I,J)=AK(I,J)+B1(I,K)*B(K,J)                                    FMK01260
      DO 700 I=1,N                                                      FMK01270
      DO 700 J=I,N                                                      FMK01280
      AK(I,J)=((AK(I,J)+AK(J,I))/2.0D0)*DTJ                             FMK01290
  700 AK(J,I)=AK(I,J)                                                   FMK01300
      RETURN                                                            FMK01310
      END                                                               FMK01320
```

Appendix 2. Codes for the generation of three-dimensional elements

```
      SUBROUTINE CUB2(STK,AK,B,N)                                       C2B0CC1C
C
CCC   CUB2 IS A NUMERICAL INTEGRATION SUBROUTINE THAT FORMS THE STIFFNESS
CCC   MATRIX FOR A LINEAR THREE DIMENSIONAL ISOPARAMETRIC ELEMENT.
CCC   TWO GAUSS ORDINATES ARE USED.
CCC
CCC   THE SUBROUTINE IS CALLED WITH THE FOLLOWING STATEMENT:
CCC   CALL CUB2(STK,AK,B,N)
CCC   THE CALLING PROGRAM MUST CONTAIN THE FOLLOWING COMMON STATEMENT:
CCC   COMMON COORD(8,3),ELAST(6,6)
CCC        STK          IS AN ARRAY DIMENSIONED NXN THAT WILL CONTAIN THE
CCC                     STIFFNESS MATRIX UPON RETURN TO THE CALLING PROGRAM.
CCC        AK           IS AN ARRAY DIMENSIONED NXN REQUIRED IN FORMK
CCC        B            IS AN ARRAY DIMENSIONED 6XN REQUIRED IN FORMK
CCC        N            IS THE NUMBER OF NODAL COORDINATES, 24 IN THIS CASE.
C
      IMPLICIT REAL*8(A-H,O-Z)                                          C2B0CC2C
      COMMON COORD(8,3),ELAST(6,6)                                      C2B0CC2S
      DIMENSION AK(N,N),STK(N,N),B(6,N)                                 C2B0CC3C
      DIMENSION XI(2)                                                   C2B0CC4C
      DATA XI/0.57735026918962582,-0.57735026918962582/                 C2B0CC5C
      DO 100 I=1,N                                                      C2B0CC6C
      DO 100 J=1,N                                                      C2B0CC7C
  100 STK(I,J)=0.0D0                                                    C2B0CC8C
      DO 200 I=1,2                                                      C2B0CC9C
      X=XI(I)                                                           C2B0CC10C
      DO 200 J=1,2                                                      C2B0CC11C
      Y=XI(J)                                                           C2B0CC12C
      DO 200 K=1,2                                                      C2B0CC13C
      Z=XI(K)                                                           C2B0CC14C
      CALL FORMK (AK,X,Y,Z,B,N)                                         C2B0CC15C
      DO 200 L=1,N                                                      C2B0CC16C
      DO 200 M=1,N                                                      C2B0CC17C
  200 STK(L,M)=STK(L,M)+(AK(L,M)/1000.000)                              C2B0CC18C
      DO 300 I=1,N                                                      C2B0CC19C
      DO 300 J=I,N                                                      C2B0CC20C
      STK(I,J)=(STK(I,J)+STK(J,I))*0.5D0                                C2B0CC21C
  300 STK(J,I)=STK(I,J)                                                 C2B0CC22C
      RETURN                                                            C2B0CC23C
      END                                                               C2B0CC24C
C
```

```
      SUBROUTINE FORMK (AK,X,Y,Z,B,N)                                   FMKLC01C
C
C
      IMPLICIT REAL*8 (A-H,O-Z)                                         FMKLC02C
      COMMON COORD(8,3),ELAST(6,6)                                      FMKLC03C
      DIMENSION AJ(3,3),DNX(3,8),W1(3,8),B(6,N),B1(24,6),AK(N,N),       FMKLC04C
     1CORDG(8,3),AJIN(3,3)                                              FMKLC05C
      DATA CORDG/1.0DC,-1.0DC,-1.0DC,1.0DC,1.0DC,-1.0DC,-1.0DC,1.0DC,   FMKLC06C
     11.0DC,1.0DC,-1.0DC,-1.0DC,-1.0DC,1.0DC,1.0DC,-1.0DC,-1.0DC,1.0DC, FMKLC07C
     21.0DC,1.0DC,-1.0DC,-1.0DC,-1.0DC,-1.0DC/                          FMKLC08C
      DF(X,Y,Z,X1,Y1,Z1)=X1*(1.0D0+Y*Y1)*(1.0D0+Z*Z1)/8.0D0             FMKLC09C
      DO 10 I=1,6                                                       FMKLC10C
      DO 10 J=1,N                                                       FMKLC11C
   10 B(I,J)=0.0D0                                                      FMKLC12C
      DO 100 I=1,8                                                      FMKLC13C
      X1=CORDG(I,1)                                                     FMKLC14C
      Y1=CORDG(I,2)                                                     FMKLC15C
      Z1=CORDG(I,3)                                                     FMKLC16C
      W1(1,I)=DF(X,Y,Z,X1,Y1,Z1)                                        FMKLC17C
      W1(2,I)=DF(Y,X,Z,Y1,X1,Z1)                                        FMKLC18C
  100 W1(3,I)=DF(Z,Y,X,Z1,Y1,X1)                                        FMKLC19C
      NPT=N/3                                                           FMKLC20C
      DO 500 I=1,3                                                      FMKLC21C
      DO 500 J=1,3                                                      FMKLC22C
      AJ(I,J)=0.0D0                                                     FMKLC23C
      DO 500 K=1,NPT                                                    FMKLC24C
  500 AJ(I,J)=AJ(I,J)+W1(I,K)*COORD(K,J)                                FMKLC25C
      DTJ=AJ(1,1)*AJ(2,2)*AJ(3,3)+AJ(1,2)*AJ(2,3)*AJ(3,1)+AJ(1,3)*AJ(2,1FMKLC26C
     1)*AJ(3,2)-AJ(3,1)*AJ(2,2)*AJ(1,3)-AJ(3,2)*AJ(2,3)*AJ(1,1)-AJ(3,3)*FMKLC27C
     2AJ(2,1)*AJ(1,2)                                                   FMKLC28C
      AJIN(1,1)= (AJ(2,2)*AJ(3,3)-AJ(3,2)*AJ(2,3))/DTJ                  FMKLC29C
      AJIN(2,1)=-(AJ(2,1)*AJ(3,3)-AJ(3,1)*AJ(2,3))/DTJ                  FMKLC30C
      AJIN(3,1)= (AJ(2,1)*AJ(3,2)-AJ(3,1)*AJ(2,2))/DTJ                  FMKLC31C
      AJIN(1,2)=-(AJ(1,2)*AJ(3,3)-AJ(3,2)*AJ(1,3))/DTJ                  FMKLC32C
      AJIN(2,2)= (AJ(1,1)*AJ(3,3)-AJ(3,1)*AJ(1,3))/DTJ                  FMKLC33C
      AJIN(3,2)=-(AJ(1,1)*AJ(3,2)-AJ(3,1)*AJ(1,2))/DTJ                  FMKLC34C
      AJIN(1,3)= (AJ(1,2)*AJ(2,3)-AJ(2,2)*AJ(1,3))/DTJ                  FMKLC35C
      AJIN(2,3)=-(AJ(1,1)*AJ(2,3)-AJ(2,1)*AJ(1,3))/DTJ                  FMKLC36C
      AJIN(3,3)= (AJ(1,1)*AJ(2,2)-AJ(2,1)*AJ(1,2))/DTJ                  FMKLC37C
      DO 600 I=1,3                                                      FMKLC38C
      DO 600 J=1,NPT                                                    FMKLC39C
      DNX(I,J)=0.0D0                                                    FMKLC40C
      DO 600 K=1,3                                                      FMKLC41C
  600 DNX(I,J)=DNX(I,J)+AJIN(I,K)*W1(K,J)                               FMKLC42C
      DO 700 I=1,NPT                                                    FMKLC43C
      I1=3*I-2                                                          FMKLC44C
      I2=I1+1                                                           FMKLC45C
      I3=I2+1                                                           FMKLC46C
```

```
      B(1,I1)=DNX(1,I)                                                  EMKL C470
      B(2,I2)=DNX(2,I)                                                  EMKL C480
      B(3,I3)=DNX(3,I)                                                  EMKL C490
      B(4,I1)=DNX(2,I)                                                  EMKL C500
      B(4,I2)=DNX(1,I)                                                  EMKL C510
      B(5,I2)=DNX(3,I)                                                  EMKL C520
      B(5,I3)=DNX(2,I)                                                  EMKL C530
      B(6,I3)=DNX(1,I)                                                  EMKL C540
  7CC B(6,I1)=DNX(3,I)                                                  EMKL C550
      DO 8CC I=1,N                                                      EMKL C560
      DO 8CC J=1,6                                                      EMKL C570
      B1(I,J)=C.CDO                                                     EMKL C580
      DO 8CC K=1,6                                                      EMKL C590
  8CC B1(I,J)=B1(I,J)+B(K,I)*ELAST(K,J)                                 EMKL C6CC
      DO 9CC I=1,N                                                      EMKL C610
      DO 9CC J=1,N                                                      EMKL C620
      AK(I,J)=C.CDO                                                     EMKL C630
      DO 9CC K=1,6                                                      EMKL C640
  9CC AK(I,J)=AK(I,J)+B1(I,K)*B(K,J)                                    EMKL C650
      WRITE(6,6CCC) DTJ                                                 EMKL C660
 6CCC FORMAT(5X,1D25.16)                                                EMKL C670
      DO 1CCC I=1,N                                                     EMKL C680
      DO 1CCC J=I,N                                                     EMKL C690
      AK(I,J)=((AK(I,J)+AK(J,I))/2.0DC)*DTJ                             EMKL C700
 1CCC AK(J,I)=AK(I,J)                                                   EMKL C710
      RETURN                                                            EMKL C720
      END                                                               EMKL C730
```

```
      SUBROUTINE CUB4(STK,AK,B,N)                                       C4POCC10
C
C     CUB4 IS A NUMERICAL INTEGRATION SUBROUTINE THAT FORMS THE STIFFNESS
C     MATRIX FOR A QUADRATIC THREE DIMENSIONAL ISOPARAMETRIC ELEMENT.
C     FOUR GAUSS ORDINATES ARE USED.
C
C     THE SUBROUTINE IS CALLED WITH THE FOLLOWING STATEMENT:
C     CALL CUB4(STK,AK,B,N)
C     THE CALLING PROGRAM MUST CONTAIN THE FOLLOWING COMMON STATEMENT:
C     COMMON COORD(20,3),ELAST(6,6)
C          STK          IS AN ARRAY DIMENSIONED NXN THAT WILL CONTAIN THE
C                       STIFFNESS MATRIX UPON RETURN TO THE CALLING PROGRAM.
C          AK           IS AN ARRAY DIMENSIONED NXN REQUIRED IN FORMK
C          B            IS AN ARRAY DIMENSIONED 6XN REQUIRED IN FORMK
C          N            IS THE NUMBER OF NODAL COORDINATES, 60 IN THIS CASE.
C
      IMPLICIT REAL*8(A-H,O-Z)                                          C4POCC20
      COMMON COORD(20,3),ELAST(6,6)                                     C4POCC30
      DIMENSION AK(N,N),STK(N,N),B(6,N)                                 C4POCC40
      DIMENSION XI(4),AI(4),AIA(4,4,4)                                  C4POCC50
      DATA XI/0.8611363115940526, 0.3399810435848563,                   C4POCC60
     1       -0.3399810435848563,-0.8611363115940526/                   C4POCC70
      DATA AI/0.3478548451374539,0.6521451548625461,                    C4POCC80
     1        0.6521451548625461,0.3478548451374539/                    C4POCC90
      DO 100 I=1,N                                                      C4POCC100
      DO 100 J=1,N                                                      C4POCC110
  100 STK(I,J)=0.000                                                    C4POCC120
      DO 150 I=1,4                                                      C4POCC130
      DO 150 J=1,4                                                      C4POCC140
      DO 150 K=1,4                                                      C4POCC150
  150 AIA(I,J,K)=AI(I)*AI(J)*AI(K)                                      C4POCC160
      DO 250 I=1,4                                                      C4POCC170
      X=XI(I)                                                           C4POCC180
      DO 250 J=1,4                                                      C4POCC190
      Y=XI(J)                                                           C4POCC200
      DO 250 K=1,4                                                      C4POCC210
      Z=XI(K)                                                           C4POCC220
      CALL FORMK(X,Y,Z)                                                 C4POCC230
      DO 200 L=1,N                                                      C4POCC240
      DO 200 M=1,N                                                      C4POCC250
  200 STK(L,M)=STK(L,M)+AIA(I,J,K)*AK(L,M)                              C4POCC260
  250 CONTINUE                                                          C4POCC270
      DO 300 I=1,N                                                      C4POCC280
      DO 300 J=I,N                                                      C4POCC290
      STK(I,J)=(STK(I,J)+STK(J,I))*0.500                                C4POCC300
  300 STK(J,I)=STK(I,J)                                                 C4POCC310
```

```
      RETURN                                                            C4BOC370
      END                                                               C4BOC380
C
C
C
      SUBROUTINE FORMK(AK,X,Y,Z,B,N)                                    FMKQCC10
C
      IMPLICIT REAL*8(A-H,O-Z)                                          FMKQCC20
      COMMON COORD(20,3),ELAST(6,6)                                     FMKQCC30
      DIMENSION AJ(3,3),AJIN(3,3),DNX(3,32),W1(3,32),B(6,N),B1(60,6),   FMKQCC40
     1AK(N,N),CORDG(20,3)                                               FMKQCC50
      DATA CORDG/1.CDO,C.CDO,-1.CDO,-1.0DO,-1.0DO,0.0DO,1.0DO,1.0DO,    FMKQCC60
     11.0DO,-1.0DO,-1.0DO,1.0DO,1.0DO,0.0DO,-1.0DO,-1.0DO,-1.0DO,0.0DO, FMKQCC70
     21.CDO,1.CDO,1.0DO,1.CDO,1.0DO,C.CDO,-1.0DO,-1.0DO,-1.0DO,0.0DO,   FMKQCC80
     31.CDO,1.CDO,-1.0DO,-1.CDO,1.0DO,1.0DO,1.0DO,0.0DO,-1.0DO,-1.0DO,  FMKQCC90
     4-1.0DO,C.CDO,1.0DO,1.0DO,1.0DO,1.0DO,1.0DO,1.0DO,1.0DO,1.0DO,     FMKQCD00
     5C.CDO,C.ODO,C.0DO,C.CDO,-1.CDO,-1.0DO,-1.0DO,-1.0DO,-1.0DO,-1.0DO,FMKQCD10
     6-1.CDO,-1.CDO/                                                    FMKQCD20
      DF(C,D,E,X1,Y1,Z1)=X1*(1.0DO+D*Y1)*(1.0DO+E*Z1)*(2.0DO*C*X1+D*Y1+ FMKQCD30
     1E*Z1-1.CDO)/8.0DO                                                 FMKQCD40
      D2(C,D,E,   Y1,Z1)=-C*(1.0DO+D*Y1)*(1.0DO+E*Z1)/2.0DO             FMKQCD50
      D4(C,   E,   Y1,Z1)=(1.CDO-C*C)*Y1*(1.0DO+E*Z1)/4.0DO             FMKQCD60
      DO 10 I=1,6                                                       FMKQCD70
      DO 10 J=1,N                                                       FMKQCD80
   10 B(I,J)=C.CDO                                                      FMKQCD90
      DO 100 I=1,2                                                      FMKQCE00
      II=12*(I-1)+1                                                     FMKQCE10
      IT=II+6                                                           FMKQCE20
      DO 100 J=II,IT,2                                                  FMKQCE30
      X1=CORDG(J,1)                                                     FMKQCE40
      Y1=CORDG(J,2)                                                     FMKQCE50
      Z1=CORDG(J,3)                                                     FMKQCE60
      W1(1,J)=DF(X,Y,Z,X1,Y1,Z1)                                        FMKQCE70
      W1(2,J)=DF(Y,Z,X,Y1,Z1,X1)                                        FMKQCE80
  100 W1(3,J)=DF(Z,X,Y,Z1,X1,Y1)                                        FMKQCE90
C     MID POINTS X1=0.0                                                 FMKQCF00
      DO 200 K=1,2                                                      FMKQCF10
      II=(K-1)*12+2                                                     FMKQCF20
      IT=II+4                                                           FMKQCF30
      DO 200 L=II,IT,4                                                  FMKQCF40
      X1=CORDG(L,1)                                                     FMKQCF50
      Y1=CORDG(L,2)                                                     FMKQCF60
      Z1=CORDG(L,3)                                                     FMKQCF70
      W1(1,L)=D2(X,Y,Z,   Y1,Z1)                                        FMKQCF80
      W1(2,L)=D4(X,   Z,   Y1,Z1)                                       FMKQCF90
  200 W1(3,L)=D4(X,   Y,   Z1,Y1)                                       FMKQCG00
      DO 300 K=1,2                                                      FMKQCG10
```

```
      II=(K-1)*12+4                                                                  FMKQC490
      IT=II+4                                                                        FMKQC500
      DO 300 L=II,IT,4                                                               FMKQC510
      X1=COORDG(L,1)                                                                 FMKQC520
      Y1=COORDG(L,2)                                                                 FMKQC530
      Z1=COORDG(L,3)                                                                 FMKQC540
      W1(1,L)=D4(Y,  Z,    X1,Z1)                                                    FMKQC550
      W1(2,L)=D2(Y,Z,X,    Z1,X1)                                                    FMKQC560
  300 W1(3,L)=D4(Y,  X,    Z1,X1)                                                    FMKQC570
      DO 400 L=9,12                                                                  FMKQC580
      X1=COORDG(L,1)                                                                 FMKQC590
      Y1=COORDG(L,2)                                                                 FMKQC600
      Z1=COORDG(L,3)                                                                 FMKQC610
      W1(1,L)=D4(Z,  Y,    X1,Y1)                                                    FMKQC620
      W1(2,L)=D4(Z,  X,    Y1,X1)                                                    FMKQC630
  400 W1(3,L)=D2(Z,X,Y,    X1,Y1)                                                    FMKQC640
      DO 500 I=1,3                                                                   FMKQC650
      DO 500 J=1,3                                                                   FMKQC660
      AJ(I,J)=0.000                                                                  FMKQC670
      DO 500 K=1,NPT                                                                 FMKQC680
  500 AJ(I,J)=AJ(I,J)+W1(I,K)*COORD(K,J)                                             FMKQC690
      DTJ=AJ(1,1)*AJ(2,2)*AJ(3,3)+AJ(1,2)*AJ(2,3)*AJ(3,1)+AJ(1,3)*AJ(2,1FMKQC700
     1)*AJ(3,2)-AJ(3,1)*AJ(2,2)*AJ(1,3)-AJ(3,2)*AJ(2,3)*AJ(1,1)-AJ(3,3)*FMKQC710
     2AJ(2,1)*AJ(1,2)                                                                FMKQC720
      AJIN(1,1)= (AJ(2,2)*AJ(3,3)-AJ(3,2)*AJ(2,3))/DTJ                               FMKQC730
      AJIN(2,1)=-(AJ(2,1)*AJ(3,3)-AJ(3,1)*AJ(2,3))/DTJ                               FMKQC740
      AJIN(3,1)= (AJ(2,1)*AJ(3,2)-AJ(3,1)*AJ(2,2))/DTJ                               FMKQC750
      AJIN(1,2)=-(AJ(1,2)*AJ(3,3)-AJ(3,2)*AJ(1,3))/DTJ                               FMKQC760
      AJIN(2,2)= (AJ(1,1)*AJ(3,3)-AJ(3,1)*AJ(1,3))/DTJ                               FMKQC770
      AJIN(3,2)=-(AJ(1,1)*AJ(3,2)-AJ(3,1)*AJ(1,2))/DTJ                               FMKQC780
      AJIN(1,3)= (AJ(1,2)*AJ(2,3)-AJ(2,2)*AJ(1,3))/DTJ                               FMKQC790
      AJIN(2,3)=-(AJ(1,1)*AJ(2,3)-AJ(2,1)*AJ(1,3))/DTJ                               FMKQC800
      AJIN(3,3)= (AJ(1,1)*AJ(2,2)-AJ(2,1)*AJ(1,2))/DTJ                               FMKQC810
      DO 600 I=1,3                                                                   FMKQC820
      DO 600 J=1,NPT                                                                 FMKQC830
      DNX(I,J)=0.000                                                                 FMKQC840
      DO 600 K=1,3                                                                   FMKQC850
  600 DNX(I,J)=DNX(I,J)+AJIN(I,K)*W1(K,J)                                            FMKQC860
      DO 700 I=1,NPT                                                                 FMKQC870
      I1=3*I-2                                                                       FMKQC880
      I2=I1+1                                                                        FMKQC890
      I3=I2+1                                                                        FMKQC900
      B(1,I1)=DNX(1,I)                                                               FMKQC910
      B(2,I2)=DNX(2,I)                                                               FMKQC920
      B(3,I3)=DNX(3,I)                                                               FMKQC930
      B(4,I1)=DNX(2,I)                                                               FMKQC940
      B(4,I2)=DNX(1,I)                                                               FMKQC950
      B(5,I2)=DNX(3,I)                                                               FMKQC960
```

```
      B(5,I3)=DNX(2,I)                                                  FMKQ0970
      B(6,I3)=DNX(1,I)                                                  FMKQ0980
  700 B(6,I1)=DNX(3,I)                                                  FMKQ0990
      DO 800 I=1,N                                                      FMKQ1000
      DO 800 J=1,6                                                      FMKQ1010
      B1(I,J)=0.0D0                                                     FMKQ1020
      DO 800 K=1,6                                                      FMKQ1030
  800 B1(I,J)=B1(I,J)+B(K,I)*ELAST(K,J)                                 FMKQ1040
      DO 900 I=1,N                                                      FMKQ1050
      DO 900 J=1,N                                                      FMKQ1060
      AK(I,J)=0.0D0                                                     FMKQ1070
      DO 900 K=1,6                                                      FMKQ1080
  900 AK(I,J)=AK(I,J)+B1(I,K)*B(K,J)                                    FMKQ1090
      DO 1000 I=1,N                                                     FMKQ1100
      DO 1000 J=I,N                                                     FMKQ1110
      AK(I,J)=((AK(I,J)+AK(J,I))/2.0D0)*DTJ                             FMKQ1120
 1000 AK(J,I)=AK(I,J)                                                   FMKQ1130
      RETURN                                                            FMKQ1140
      END                                                               FMKQ1150
```

```
      SUBROUTINE CUBS(STK,AK,B,N)                                       C5POCC1C
C
C     CUBS IS A NUMERICAL INTEGRATION SUBROUTINE THAT FORMS THE STIFFNESS
C     MATRIX FOR A CUBIC THREE DIMENSIONAL ISOPARAMETRIC ELEMENT.
C     FIVE GAUSS ORDINATES ARE USED.
C
C     THE SUBROUTINE IS CALLED WITH THE FOLLOWING STATEMENT:
C     CALL CUBS(STK,AK,B,N)
C     THE CALLING PROGRAM MUST CONTAIN THE FOLLOWING COMMON STATEMENT:
C     COMMON COORD(32,3),ELAST(6,6)
C          STK          IS AN ARRAY DIMENSIONED NXN THAT WILL CONTAIN THE
C                       STIFFNESS MATRIX UPON RETURN TO THE CALLING PROGRAM.
C          AK           IS AN ARRAY DIMENSIONED NXN REQUIRED IN FORMK
C          B            IS AN ARRAY DIMENSIONED 6XN REQUIRED IN FORMK
C          N            IS THE NUMBER OF NODAL COORDINATES, 96 IN THIS CASE.
C
      IMPLICIT REAL*8(A-H,O-Z)                                          C5POCC2C
      COMMON COORD(32,3),ELAST(6,6)                                     C5POCC3C
      DIMENSION AK(N,N),STK(N,N),B(6,N)                                 C5POCC4C
      DIMENSION XI(5),AI(5),AIA(5,5,5)                                  C5POCC5C
      DATA XI/C.9061798459386640,0.5384693101056831,0.0D0,-0.5384693101C5POCC6C
     1569831,-C.9C61798459386640/                                       C5POCC7C
      DATA AI/0.2369268850561891,C.4786286704993665,0.5688888888888889,C5POCC8C
     1.4786286704993665,0.2369268850561891/                             C5POCC9C
      DO 100 I=1,N                                                      C5POC1CC
      DO 100 J=1,N                                                      C5POC11C
  100 STK(I,J)=C.0D0                                                    C5POC12C
      DO 150 I=1,5                                                      C5POC13C
      DO 150 J=1,5                                                      C5POC14C
      DO 150 K=1,5                                                      C5POC15C
  150 AIA(I,J,K)=AI(I)*AI(J)*AI(K)                                      C5POC16C
      DO 200 I=1,5                                                      C5POC17C
      X=XI(I)                                                           C5POC18C
      DO 200 J=1,5                                                      C5POC19C
      Y=XI(J)                                                           C5POC2CC
      DO 200 K=1,5                                                      C5POC21C
      Z=XI(K)                                                           C5POC22C
      CALL FORMK(AK,X,Y,Z,B,N)                                          C5POC23C
      DO 200 L=1,N                                                      C5POC24C
      DO 200 M=1,N                                                      C5POC25C
  200 STK(L,M)=STK(L,M)+AIA(I,J,K)*AK(L,M)                              C5POC26C
      DO 300 I=1,N                                                      C5POC27C
      DO 300 J=I,N                                                      C5POC28C
      STK(I,J)=(STK(I,J)+STK(J,I))*0.5D0                                C5POC29C
  300 STK(J,I)=STK(I,J)                                                 C5POC3CC
      RETURN
```

```
      END                                                                C5E0C31C
C
C
      SUBROUTINE FORMK(AK,X,Y,Z,B,N)                                     FMKCCC1C
C
      IMPLICIT REAL*8(A-H,O-Z)                                           FMKCC020
      COMMON COORD(32,3),ELAST(6,6)                                      FMKCC030
      DIMENSION AJ(3,3),AJIN(3,3),DNX(3,32),W1(3,32),B(6,N),B1(96,6),    FMKCCC4C
     1AK(N,N),CORDG(32,3)                                                FMKCC050
      DATA CCRDG/1.0D0,0.3333333333333333,-0.3333333333333333,-1.0D0,    FMKCCC6C
     1-1.0D0,-1.0D0,-1.0D0,-0.3333333333333333,0.3333333333333333,1.0D0,FMKCC07C
     21.0D0,1.0D0,1.0D0,-1.0D0,-1.0D0,1.0D0,1.0D0,-1.0D0,-1.0D0,1.0D0,   FMKCCC8C
     31.0D0,0.3333333333333333,-0.3333333333333333,-1.0D0,-1.0D0,-1.0D0,FMKCCC9C
     4-1.0D0,-0.3333333333333333,0.3333333333333333,1.0D0,1.0D0,1.0D0,   FMKCC100
     51.0D0,1.0D0,1.0D0,1.0D0,0.3333333333333333,-0.3333333333333333,    FMKCC110
     6-1.0D0,-1.0D0,-1.0D0,-1.0D0,-0.3333333333333333,0.3333333333333333FMKCC12C
     7,1.0D0,1.0D0,-1.0D0,-1.0D0,1.0D0,1.0D0,-1.0D0,-1.0D0,1.0D0,1.0D0,  FMKCC130
     81.0D0,1.0D0,0.3333333333333333,-0.3333333333333333,-1.0D0,-1.0D0,  FMKCC140
     9-1.0D0,-1.0D0,-0.3333333333333333,0.3333333333333333,1.0D0,1.0D0,  FMKCC15C
     A1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,       FMKCC160
     B0.3333333333333333,0.3333333333333333,0.3333333333333333,0.3333333FMKCC170
     C333333333,-0.3333333333333333,-0.3333333333333333 ,-0.333333333333FMKCC180
     D3333,-0.3333333333333333,-1.0D0,-1.0D0,-1.0D0,-1.0D0,-1.0D0,-1.0D0FMKCC190
     E,-1.0D0,-1.0D0,-1.0D0,-1.0D0,-1.0D0,-1.0D0,-1.0D0/                 FMKCC2CC
      DFX(X,Y,Z,X1,Y1,Z1)=(1.0D0+Y*Y1)*(1.0D0+Z*Z1)*(9.0D0*X1*(X*X+Y*Y+ FMKCC21C
     1Z*Z)-19.0D0*X1+18.0D0*X+18.0D0*X*X*X1)/64.0D0                      FMKCC220
      DX(X,Y,Z,X1,Y1,Z1)=9.0D0*(1.0D0+Y*Y1)*(1.0D0+Z*Z1)*(-2.0D0*X-27.0DFMKCC23C
     1C*X*X*X1+9.0D0*X1)/64.0D0                                          FMKCC240
      DY(X,  Z,X1,Y1,Z1)=9.0D0*(1.0D0-X*X)*(1.0D0+9.0D0*X*X1)*Y1*(1.0D0+FMKCC25C
     1Z*Z1)/64.0D0                                                       FMKCC260
      DZ(X,Y,  X1,Y1,Z1)=DY(X,  Y,X1,Z1,Y1)                             FMKCC27C
      DO 1C I=1,6                                                        FMKCC280
      DO 1C J=1,N                                                        FMKCC290
   1C B(I,J)=0.0D0                                                       FMKCC300
      DO 100 I=1,2                                                       FMKCC310
      II=20*(I-1)+1                                                      FMKCC320
      IT=II+9                                                            FMKCC330
      DO 100 J=II,IT,3                                                   FMKCC340
      X1=CORDG(J,1)                                                      FMKCC350
      Y1=CORDG(J,2)                                                      FMKCC360
      Z1=CORDG(J,3)                                                      FMKCC370
      W1(1,J)=DFX(X,Y,Z,X1,Y1,Z1)                                        FMKCC380
      W1(2,J)=DFX(Y,X,Z,Y1,X1,Z1)                                        FMKCC390
  100 W1(3,J)=DFX(Z,X,Y,Z1,X1,Y1)                                        FMKCC400
C    MIDDLE POINTS X=(+OR-)1/3                                           FMKCC41C
      DO 200 K=1,2                                                       FMKCC420
```

```
      IK=20*(K-1)                                                       FMKCC430
      DO 200 I=1,2                                                      FMKCC440
      II=6*(I-1)+2+IK                                                   FMKCC450
      IT=II+1                                                           FMKCC460
      DO 200 J=II,IT                                                    FMKCC470
      X1=CORDG(J,1)                                                     FMKCC480
      Y1=CORDG(J,2)                                                     FMKCC490
      Z1=CORDG(J,3)                                                     FMKCC500
      W1(1,J)=DX(X,Y,Z,X1,Y1,Z1)                                        FMKCC510
      W1(2,J)=DY(X,  Z,X1,Y1,Z1)                                        FMKCC520
  200 W1(3,J)=DZ(X,Y,  X1,Y1,Z1)                                        FMKCC530
C     MIDDLE POINTS Y=(+OR-)1/3                                         FMKCC540
      DO 300 K=1,2                                                      FMKCC550
      IK=20*(K-1)                                                       FMKCC560
      DO 300 I=1,2                                                      FMKCC570
      II=6*(I-1)+5+IK                                                   FMKCC580
      IT=II+1                                                           FMKCC590
      DO 300 J=II,IT                                                    FMKCC600
      X1=CORDG(J,1)                                                     FMKCC610
      Y1=CORDG(J,2)                                                     FMKCC620
      Z1=CORDG(J,3)                                                     FMKCC630
      W1(1,J)=DY(Y,  Z,Y1,X1,Z1)                                        FMKCC640
      W1(2,J)=DX(Y,X,Z,Y1,X1,Z1)                                        FMKCC650
  300 W1(3,J)=DZ(Y,X,  Y1,X1,Z1)                                        FMKCC660
C MIDDLE POINTS Z=(+OR-)1/3                                             FMKCC670
      DO 400 K=1,2                                                      FMKCC680
      IK=4*(K-1)                                                        FMKCC690
      II=13+IK                                                          FMKCC700
      IT=II+3                                                           FMKCC710
      DO 400 J=II,IT                                                    FMKCC720
      X1=CORDG(J,1)                                                     FMKCC730
      Y1=CORDG(J,2)                                                     FMKCC740
      Z1=CORDG(J,3)                                                     FMKCC750
      W1(1,J)=DZ(Z,Y,  Z1,Y1,X1)                                        FMKCC760
      W1(2,J)=DY(Z,  X,Z1,Y1,X1)                                        FMKCC770
  400 W1(3,J)=DX(Z,Y,X,Z1,Y1,X1)                                        FMKCC780
      NPT=N/3                                                           FMKCC790
      DO 500 I=1,3                                                      FMKCC800
      DO 500 J=1,3                                                      FMKCC810
      AJ(I,J)=0.000                                                     FMKCC820
      DO 500 K=1,NPT                                                    FMKCC830
  500 AJ(I,J)=AJ(I,J)+W1(I,K)*COORD(K,J)                                FMKCC840
      DTJ=AJ(1,1)*AJ(2,2)*AJ(3,3)+AJ(1,2)*AJ(2,3)*AJ(3,1)+AJ(1,3)*AJ(2,1FMKCC850
     1)*AJ(3,2)-AJ(3,1)*AJ(2,2)*AJ(1,3)-AJ(3,2)*AJ(2,3)*AJ(1,1)-AJ(3,3)*FMKCC860
     2AJ(2,1)*AJ(1,2)                                                   FMKCC870
      AJIN(1,1)= (AJ(2,2)*AJ(3,3)-AJ(3,2)*AJ(2,3))/DTJ                  FMKCC880
      AJIN(2,1)=-(AJ(2,1)*AJ(3,3)-AJ(3,1)*AJ(2,3))/DTJ                  FMKCC890
      AJIN(3,1)= (AJ(2,1)*AJ(3,2)-AJ(3,1)*AJ(2,2))/DTJ                  FMKCC900
```

```
      AJIN(1,2)=-(AJ(1,2)*AJ(3,3)-AJ(3,2)*AJ(1,3))/DTJ                  FMKCC910
      AJIN(2,2)= (AJ(1,1)*AJ(3,3)-AJ(3,1)*AJ(1,3))/DTJ                  FMKCC920
      AJIN(3,2)=-(AJ(1,1)*AJ(3,2)-AJ(3,1)*AJ(1,2))/DTJ                  FMKCC930
      AJIN(1,3)= (AJ(1,2)*AJ(2,3)-AJ(2,2)*AJ(1,3))/DTJ                  FMKCC940
      AJIN(2,3)=-(AJ(1,1)*AJ(2,3)-AJ(2,1)*AJ(1,3))/DTJ                  FMKCC950
      AJIN(3,3)= (AJ(1,1)*AJ(2,2)-AJ(2,1)*AJ(1,2))/DTJ                  FMKCC960
      DO 60C  I=1,3                                                     FMKCC970
      DO 6CC J=1,NPT                                                    FMKCC980
      DNX(I,J)=C.0D0                                                    FMKCC990
      DO 6CC K=1,3                                                      FMKC1000
  60C DNX(I,J)=DNX(I,J)+AJIN(I,K)*W1(K,J)                               FMKC1010
      DO 700  I=1,NPT                                                   FMKC1020
      I1=3*I-2                                                          FMKC1030
      I2=I1+1                                                           FMKC1040
      I3=I2+1                                                           FMKC1050
      B(1,I1)=DNX(1,I)                                                  FMKC1060
      B(2,I2)=DNX(2,I)                                                  FMKC1070
      B(3,I3)=DNX(3,I)                                                  FMKC1080
      B(4,I1)=DNX(2,I)                                                  FMKC1090
      B(4,I2)=DNX(1,I)                                                  FMKC1100
      B(5,I2)=DNX(3,I)                                                  FMKC1110
      B(5,I3)=DNX(2,I)                                                  FMKC1120
      B(6,I3)=DNX(1,I)                                                  FMKC1130
  700 B(6,I1)=DNX(3,I)                                                  FMKC1140
      DO 800  I=1,N                                                     FMKC1150
      DO 8CC J=1,6                                                      FMKC1160
      B1(I,J)=C.0D0                                                     FMKC1170
      DO 800 K=1,6                                                      FMKC1180
  80C B1(I,J)=B1(I,J)+B(K,I)*ELAST(K,J)                                 FMKC1190
      DO 900  I=1,N                                                     FMKC1200
      DO 9C0 J=1,N                                                      FMKC1210
      AK(I,J)=0.0D0                                                     FMKC1220
      DO 900 K=1,6                                                      FMKC1230
  90C AK(I,J)=AK(I,J)+B1(I,K)*B(K,J)                                    FMKC1240
      WRITE(6,6CCC) DTJ                                                 FMKC1250
 6CCC FORMAT(5X,1D25.16)                                                FMKC1260
      DO 100C  I=1,N                                                    FMKC1270
      DO 100C J=I,N                                                     FMKC1280
      AK(I,J)=((AK(I,J)+AK(J,I))/2.0D0)*DTJ                             FMKC1290
 100C AK(J,I)=AK(I,J)                                                   FMKC1300
      RETURN                                                            FMKC1310
      END                                                               FMKC1320
```

Appendix 3. Listing for an in-core band solver

```
      SUBROUTINE SYMSOL(A,B,C,NN,MM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(NN,MM),B(NN),C(MM)
C
CC       A     IS THE COEFFICIENT MATRIX
CC       B     IS THE RIGHT HAND MEMBER
CC       C     IS A WORKING SPACE NEEDED IN THE SUBROUTINE
CC       NN    IS THE TOTAL NUMBER OF EQUATIONS
CC       MM    IS THE HALF-BANDWIDTH OF THE SYSTEM
C
      N=0
  100 N=N+1
C
CCCC     REDUCE N TH EQUATION
C
CCCCC          1. DIVIDE RIGHT SIDE BY DIAGONAL ELEMENT
C
      B(N)=B(N)/A(N,1)
C
CCC            2. CHECK FOR LAST EQUATION
C
      IF(N-NN) 150,300,150
C
CCC            3. DIVIDE N TH EQUATION BY DIAGONAL ELEMENT
C
  150 DO 200 K=2,MM
      C(K)=A(N,K)
  200 A(N,K)=A(N,K)/A(N,1)
C
CCC            4. REDUCE REMAINING EQUATIONS
C
      DO 260 L=2,MM
      I=N+L-1
      IF(NN-I) 260,240,240
  240 J=0
      DO 250 K=L,MM
      J=J+1
  250 A(I,J)=A(I,J)-C(L)*A(N,K)
      B(I)=B(I)-C(L)*B(N)
  260 CONTINUE
      GO TO 100
C
CCC      BACK SUBSTITUTION
C
  300 N=N-1
C
```

```
C               1. CHECK FOR FIRST EQUATION
C
      IF(N) 350,500,350
C
C               2. CALCULATE UNKNOWNS  B(N)
C
  350 DO 400 K=2,MM
      L=N+K-1
      IF(NN-L) 400,370,370
  370 B(N)=B(N)-A(N,K)*B(L)
  400 CONTINUE
      GO TO 300
  500 RETURN
      END
```

Appendix 4. Listing for the Block Solver

```
      SUBROUTINE SOLVE(NN,MM,NS)                                        SLV00100
      IMPLICIT REAL*8(A-H,O-Z)                                          SLV00105
C*********************************************************************
C
C                        CODED BY: GILLES CANTIN, DECEMBER 1969
C                        TESTED ON AN IBM 360/67
C
C        NN    IS THE NUMBER OF BLOCKS PER COLUMN
C        MM    IS THE NUMBER OF BLOCKS PER ROW
C        NS    IS THE SIZE OF ONE BLOCK
C        NSIZ  IS EQUAL TO NS*NS
C              IN THIS VERSION NS IS 50, TO MODIFY, THE DIMENSION
C              STATEMENT MUST BE REPUNCHED ACCORDING TO CARD ZZZZZZZZ
C              WITH THE PARAMETERS NSIZ AND NS REPLACED BY NUMBERS
C     DIMENSION AK1(NSIZ),AK2(NSIZ),AK3(NSIZ),RB1(NS),RB2(NS),RB3(NS)   ZZZZZZZZ
C
C*********************************************************************
      DIMENSION AK1(2500),AK2(2500),AK3(2500),RB1(50),RB2(50),RB3(50)   ZZZZZZZZ
      NTRK(I,J)=(I-1)*(MM+2)+J-I+1                                      SLV00110
      NCOUNT = NS*NS                                                    SLV00120
      N=0                                                               SLV00130
      KMM=MM
  100 N=N+1                                                             SLV00140
C                                                                       SLV00150
C        REDUCE BLOCK ROW  "N"                                          SLV00160
C                                                                       SLV00170
C        1.-  REDUCE R.H.S.                                             SLV00180
C                                                                       SLV00190
      NTK=NTRK(N,1)                                                     SLV00200
      NTR=N*(MM+1)                                                      SLV00210
      CALL RDDISK(NTK,AK2,NCOUNT)                                       SLV00220
      CALL SYMINV(AK2,NS,RB1,RB2,IFLG)                                  SLV00230
      IF(IFLG.EQ.1) GO TO 600                                           SLV00240
      IF(IFLG.EQ.2) WRITE(6,2000) N                                     SLV00242
 2000 FORMAT(5X,'BLOCK (',I3,', 1) IS NEARLY SINGULAR')                 SLV00244
      CALL RDDISK(NTR,RB1,NS)                                           SLV00250
      CALL MULT(AK2,RB1,RB2,NS,NS,1)                                    SLV00260
      CALL WRDISK(NTR,RB2,NS)                                           SLV00270
C                                                                       SLV00280
C        2.-  CHECK FOR LAST ROW OF BLOCKS                              SLV00290
C                                                                       SLV00300
      IF(N.EQ.NN) GO TO 300                                             SLV00310
C                                                                       SLV00320
C        3.-  REDUCE BLOCKS IN ROW "N"                                  SLV00330
C                                                                       SLV00340
      IF(N.GT.(NN-MM+1)) KMM=KMM-1
```

```
      DO 200 K=2,KMM
      NTK=NTRK(N,K)                                                    SLV00360
      CALL RDDISK(NTK,AK1,NCOUNT)                                      SLV00370
      CALL MULT(AK2,AK1,AK3,NS,NS,NS)                                  SLV00380
      CALL WRDISK(NTK,AK3,NCOUNT)                                      SLV00390
      DO 150 I=1,NS                                                    SLV00391
      II=(I-1)*NS                                                      SLV00392
      DO 150 J=1,NS                                                    SLV00393
      IL=II+J                                                          SLV00394
      IR=I+(J-1)*NS                                                    SLV00395
  150 AK3(IL)=AK1(IR)                                                  SLV00396
      NTK=NTRK(NN,K)                                                   SLV00400
      CALL WRDISK(NTK,AK3,NCOUNT)                                      SLV00410
  200 CONTINUE                                                         SLV00420
C                                                                      SLV00430
CC       4.-  REDUCE REMAINING ROWS OF BLOCKS                          SLV00440
C                                                                      SLV00450
      DO 260 L=2,KMM
      I=N+L-1                                                          SLV00470
      IF(I.GT.NN) GO TO 260                                            SLV00480
      J=0                                                              SLV00490
      NTK=NTRK(NN,L)                                                   SLV00500
      CALL RDDISK (NTK,AK2,NCOUNT)                                     SLV00510
      DO 250 K=L,KMM
      J=J+1                                                            SLV00530
      NTK=NTRK(N,K)                                                    SLV00540
      CALL RDDISK(NTK,AK1,NCOUNT)                                      SLV00550
      CALL MULT(AK2,AK1,AK3,NS,NS,NS)                                  SLV00560
      NTK=NTRK(I,J)                                                    SLV00570
      CALL RDDISK(NTK,AK1,NCOUNT)                                      SLV00580
      DO 210 I1=1,NCOUNT                                               SLV00590
  210 AK1(I1)=AK1(I1)-AK3(I1)                                          SLV00595
      CALL WRDISK(NTK,AK1,NCOUNT)                                      SLV00600
  250 CONTINUE                                                         SLV00610
      CALL MULT(AK2,RB2,RB3,NS,NS,1)                                   SLV00620
      NTR=I*(MM+1)                                                     SLV00630
      CALL RDDISK(NTR,RB1,NS)                                          SLV00640
      DO 255 I1=1,NS                                                   SLV00650
  255 RB1(I1)=RB1(I1)-RB3(I1)                                          SLV00660
      CALL WRDISK(NTR,RB1,NS)                                          SLV00700
  260 CONTINUE                                                         SLV00710
      GO TO 100                                                        SLV00720
C                                                                      SLV00730
CC       BACK SUBSTITUTION                                             SLV00740
C                                                                      SLV00750
  300 N=N-1                                                            SLV00760
C                                                                      SLV00770
C        1.-  CHECK FOR FIRST ROW OF BLOCKS                            SLV00780
```

```
C                                                                       SLV00790
      IF (N.EQ.0) GO TO 500                                             SLV00800
C                                                                       SLV00810
C        2.- CALCULATE BLOCKS OF UNKNOWNS                               SLV00820
C                                                                       SLV00830
      NT1=N*(MM+1)                                                      SLV00832
      CALL RDDISK(NT1,RB3,NS)                                           SLV00834
      DO 400 K=2,KMM
      L=N+K-1                                                           SLV00850
      IF(L.GT.NN) GO TO 400                                             SLV00860
      NTK=NTRK(N,K)                                                     SLV00870
      CALL RDDISK(NTK,AK1,NCOUNT)                                       SLV00880
      NTR=L*(MM+1)                                                      SLV00890
      CALL RDDISK(NTR,RB1,NS)                                           SLV00900
      CALL MULT(AK1,RB1,RB2,NS,NS,1)                                    SLV00910
      DO 310 K1=1,NS                                                    SLV00920
  310 RB3(K1)=RB3(K1)-RB2(K1)                                           SLV00930
  400 CONTINUE                                                          SLV00940
      CALL WRDISK(NT1,RB3,NS)                                           SLV00950
      KMM=KMM+1
      IF(KMM.GT.MM) KMM=MM
      GO TO 300                                                         SLV00970
  500 RETURN                                                            SLV00980
  600 WRITE(6,1000) N                                                   SLV00990
 1000 FORMAT(5X,'BLOCK (',I3,', 1) IS SINGULAR')                        SLV01000
      STOP                                                              SLV01010
      END                                                               SLV01020
```

```
      SUBROUTINE MULT(A,B,C,NRA,NCA,NCB)                                MLT00010
      IMPLICIT REAL*8(A-H,O-Z)                                          MLT00020
      DIMENSION A(1),B(1),C(1)                                          MLT00030
      ZRO=0.0D0                                                         MLT00040
      DO 100 I=1,NRA                                                    MLT00050
      DO 100 J=1,NCB                                                    MLT00060
      IC=I+(J-1)*NRA                                                    MLT00070
      C(IC)=ZRO                                                         MLT00080
      DO 100 K=1,NCA                                                    MLT00090
      IA=I+(K-1)*NRA                                                    MLT00100
      IB=K+(J-1)*NCA                                                    MLT00110
      C(IC)=C(IC)+A(IA)*B(IB)                                           MLT00130
  100 CONTINUE                                                          MLT00140
      RETURN                                                            MLT00150
      END                                                               MLT00160
```

```
      SUBROUTINE SYMINV(A,N,B,C,IFLG)                                   SIV00010
      IMPLICIT REAL*8(A-H,O-Z)                                          SIV00020
      DIMENSION A(1),B(1),C(1)                                          SIV00030
C*******************************************************************  SIV00040
C                                                                       SIV00050
C       SYMMETRICAL MATRIX INVERSION                                    SIV00060
C                                                                       SIV00070
C*******************************************************************  SIV00080
      IFLG=0                                                            SIV00090
      ZRO=0.0D0                                                         SIV00100
      TRACE=ZRO                                                         SIV00110
      DO 5 I=1,N                                                        SIV00120
      IAD=(I-1)*N+I                                                     SIV00130
    5 TRACE=TRACE+DABS(A(IAD))                                          SIV00140
      APZRO=TRACE*1.0D-12                                               SIV00150
      LP=N                                                              SIV00160
      DO 10  I=2,LP                                                     SIV00170
      DO 10  J=I,LP                                                     SIV00180
      ICOL=J+LP*(I-2)                                                   SIV00190
      IROW=I+(J-1)*LP-1                                                 SIV00200
      IF(A(ICOL).EQ.A(IROW)) GO TO 10                                   SIV00210
      A(ICOL)=0.5D0*(A(ICOL)+A(IROW))                                   SIV00220
      A(IROW)=A(ICOL)                                                   SIV00230
   10 CONTINUE                                                          SIV00240
      DO 20 I=1,N                                                       SIV00250
      B(I)=ZRO                                                          SIV00260
      II=(I-1)*N                                                        SIV00270
      DO 20 J=1,N                                                       SIV00280
      JJ=II+J                                                           SIV00290
   20 B(I)=B(I)+DABS(A(JJ))                                             SIV00300
      ANR=ZRO                                                           SIV00310
      DO 30 I=1,N                                                       SIV00320
   30 ANR=DMAX1(ANR,B(I))                                               SIV00330
      DO 140 I=1,N                                                      SIV00340
      NR=(I-1)*N                                                        SIV00350
C                                                                       SIV00360
      DO 100 J=1,N                                                      SIV00370
      K=NR+J                                                            SIV00380
  100 B(J)=A(K)                                                         SIV00390
      D=B(I)                                                            SIV00400
      IF(D.EQ.ZRO) GO TO 180                                            SIV00410
      IF(DABS(D).LT.APZRO) IFLG=2                                       SIV00420
      DO 110 J=1,N                                                      SIV00430
  110 C(J)=-B(J)/D                                                      SIV00440
C                                                                       SIV00450
      L=1                                                               SIV00460
```

```
      DO 130 J=1,N                                                      SIV00450
      M=L                                                               SIV00460
      DO 120 K=J,N                                                      SIV00470
      A(L)=A(L)+B(J)*C(K)                                               SIV00490
  115 CONTINUE                                                          SIV00500
      A(M)=A(L)                                                         SIV00510
      M=M+N                                                             SIV00520
  120 L=L+1                                                             SIV00530
  130 L=L+J                                                             SIV00540
C                                                                       SIV00550
      C(I)=-1.0D0/D                                                     SIV00560
      M=I                                                               SIV00570
      DO 140 J=1,N                                                      SIV00580
      K=NR+J                                                            SIV00590
      A(K)=C(J)                                                         SIV00600
      A(M)=C(J)                                                         SIV00610
  140 M=M+N                                                             SIV00620
C                                                                       SIV00630
      NS=N*N                                                            SIV00640
      DO 150 J=1,NS                                                     SIV00650
  150 A(J)=-A(J)                                                        SIV00660
C                                                                       SIV00670
      DO 160 I=1,N                                                      SIV00680
      B(I)=ZRO                                                          SIV00690
      II=(I-1)*N                                                        SIV00700
      DO 160 J=1,N                                                      SIV00710
      JJ=II+J                                                           SIV00720
  160 B(I)=B(I)+DABS(A(JJ))                                             SIV00730
      AINR=ZRO                                                          SIV00740
      DO 170 I=1,N                                                      SIV00750
  170 AINR=DMAX1(AINR,B(I))                                             SIV00760
      CNBR=ANR*AINR                                                     SIV00770
      WRITE(6,2000) CNBR                                                SIV00780
 2000 FORMAT(5X,'CONDITION NUMBER ',5X,1PD25.16)                        SIV00790
      RETURN                                                            SIV00800
  180 IFLG=1                                                            SIV00810
      RETURN                                                            SIV00820
      END                                                               SIV00830
```

```
      SUBROUTINE WRDISK(NTRACK, A,NCT)                                  RDK00010
      INTEGER LAST/0/                                                   RDK00020
      REAL*8 NAME(2)/'WRDISK','RDDISK'/                                 RDK00030
      REAL*8 A                                                          RDK00035
      DIMENSION A(1)                                                    RDK00040
C*******************************************************************
C
C        THIS IS A DIRECT ACCESS SUBROUTINE TO READ OR WRITE ON DISK
C        NS    IS THE SAME PARAMETER AS USED IN SOLVE
C        NRECORD IS THE LENGTH OF ONE RECORD,IT MUST BE 8*NS
C        NRMAX IS THE MAXIMUM NUMBER OF RECORDS IT MUST BE NN*(MM+1)*NS
C            (THE PARAMETERS NN AND MM ARE THE SAME AS IN SOLVE)
C
C        TO CHANGE THE CAPACITY OF THE SUBROUTINE, THE CARDS NUMBERED
C             ZZZ      MUST BE REPUNCHED , WITH THE PARAMETERS
C             NRMAX, NRECORD AND NS REPLACED BY NUMBERS
C     DEFINE FILE 7(NRMAX,NRECORD,E,I)                                  ZZZ00010
C     NRL=NS                                                            ZZZ00020
C     NRCM=NRMAX                                                        ZZZ00030
C1000 FORMAT (NSA8)                                                     ZZZ00040
C
C        THE MAXIMUM NUMBER OF WORDS IN A OR B MUST BE .LE. NRL*NS
C
C        THE NUMBER OF CYLINDERS REQUIRED ON AN IBM/2314 DISK PACK
C        IS  NRMAX /NC, WHERE NC IS (120,280,460) FOR NS EQUAL
C        TO: (100,50,25) RESPECTIVELY.
C
C*******************************************************************
      DEFINE FILE 7(2000,400,E,I)                                       ZZZ00010
      NRL=50                                                            ZZZ00020
      NRCM=2000                                                         ZZZ00030
 1000 FORMAT (50A8)                                                     ZZZ00040
      NRCMM=NRCM-1                                                      RDK00045
      NRCD= NRCM/NRL                                                    RDK00050
      IF(NTRACK .GT. NRCD ) GO TO 900                                   RDK00060
      IF(NTRACK .LT. 0) NTRACK =(LAST+NRL-1)/NRL                        RDK00070
      N = NTRACK-1                                                      RDK00080
      N=N*NRL+1                                                         RDK00090
      IF (NCT.GT. NRL)GO TO 50                                          RDK00100
      WRITE (7'N,1000) (A(J),J=1,NCT)                                   RDK00110
      IF (LAST .LT. I) LAST = I                                         RDK00120
      IF(LAST.GT. NRCMM)LAST=0                                          RDK00130
      RETURN                                                            RDK00140
   50 JI = NRL+1                                                        RDK00150
      WRITE (7'N,1000) (A(J),J=1,NRL)                                   RDK00160
   75 JE = JI + NRL - 1                                                 RDK00170
```

```
      IF ( JE .GE. NCT) GO TO 100                                          RDK00180
      WRITE (7'I,1000) (A(J),J=JI,JE)                                      RDK00190
      JI = JI+NRL                                                          RDK00200
      GO TO 75                                                             RDK00210
  100 WRITE (7'I,1000) (A(J),J=JI,NCT)                                     RDK00220
      IF(I.GT.LAST) LAST=I                                                 RDK00230
      IF(LAST.GT. NRCMM)LAST=0                                             RDK00240
      RETURN                                                               RDK00250
      ENTRY RDDISK(NTRACK,B,NCT)                                           RDK00260
      REAL*8 B                                                             RDK00265
      DIMENSION B(1)                                                       RDK00270
      IF(NTRACK .GT.NRCD ) GO TO 910                                       RDK00280
      N=NTRACK-1                                                           RDK00285
      N=N*NRL+1                                                            RDK00290
      IF (NCT .GT.NRL)GO TO 150                                            RDK00300
      READ (7'N,1000) (B(J), J=1,NCT)                                      RDK00310
      RETURN                                                               RDK00320
  150 READ (7'N,1000) (B(J),J=1,NRL)                                       RDK00330
      JI = NRL + 1                                                         RDK00340
  175 JE = JI + NRL - 1                                                    RDK00350
      IF (JE .GE. NCT) GO TO 200                                           RDK00360
      READ  (7'I,1000) (B(J),J=JI,JE)                                      RDK00370
      JI = JI + NRL                                                        RDK00380
      GO TO 175                                                            RDK00390
  200 READ  (7'I,1000) (B(J),J=JI,NCT)                                     RDK00400
      RETURN                                                               RDK00410
  900 K=1                                                                  RDK00430
      GO TO 920                                                            RDK00440
  910 K=2                                                                  RDK00450
  920 WRITE(6,1920)NAME(K),NRCD,NTRACK                                     RDK00460
 1920 FORMAT('0 ERROR IN CALL OF ',A6,', NTRACK TOO LARGE,(MUST BE .LT.    RDK00470
     1THAN ',I5,/,'NTRACK = ',I5)                                          RDK00480
      STOP                                                                 RDK00490
      END                                                                  RDK00500
```

REFERENCES

1. Becker, E. B., and Brisbane, J. J., "Application of the Finite Element Method to Stress Analysis of Solid Propellant Rocket Grains," U. S. Army Missile Command, Redstone Arsenal, Alabama; Rohm and Haas Co., Huntsville, Alabama. Technical report S-76; Volume 1, Nov. 18, 1965; Volume 2, part 1 and part 2, Jan. 21, 1966.

2. Felippa, C. A., "Refined Finite Element Analysis of Linear and Nonlinear Two-dimensional Structures," Structural Engineering Laboratory Report No. 66-22a, University of California, Berkeley, 1966.

3. Irons, B. M., and Zienkiewicz, O. C., "The Isoparametric Finite Element System, a New Concept in Finite Element Analysis". Research Report No. C/R/84/68, University of Wales, Swansea, England, March 1968.

4. Holand, I. and Bell, K., "Finite Element Method in Stress Analysis," Chapter 13 by Zienkiewicz, Irons, Ergatoudis, Ahmad and Scott, pp 383-435, Tapir, Technical University of Norway, Trondheim, Norway, 1969.

5. Clough, Ray W., "Comparison of Three-dimensional Finite Elements," ASCE Proceedings of the Symposium on Finite Element in Civil Engineering, School of Engineering, Vanderbilt University, Nashville Tennessee, November 13-14, 1969, pp 1-26.

6. Cantin, G., "An Equation Solver of Very Large Capacity," to be published in the International Journal for Numerical Methods in Engineering.

7. Taylor, R. L., Pister, K. S. and Goudreau, G. L., "Thermomechanical Analysis of Viscoelastic Solids," Report No. 68-7, Structures and Materials Research, Department of Civil Engineering, University of California, Berkeley, California, June 1968.

8. Irons, B. M., "A Frontal Solution Program for Finite Element Analysis," Int. J. for Numerical Methods in Engineering, Vol. 2, No. 1 (January-March 1970).

9. Klyuyev, V. V., and Kokovkin-Shcherbak, N. I., "On the Minimization of the Number of Arithmatic Operations for Solution of Linear Algebraic Systems of Equations," (translated by G. J. Tee), Computer Science Department, Technical Report CS24, Stanford University, Stanford, California (1965)

10. Forsythe, G., and Moler, C. B., "Computer Solution of Linear Algebraic Systems," Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967, pp 26, 27.

11. Brooks, D. F., and Brotton, D. M., "Computer System for Analysis of Large Frameworks," Journal of the Structural Division, ASCE, December 1967, pp 1-23.

12. Melosh, R. J., and Bamford, R. M., "Efficient Solution of Load-Deflection Equations," Journal of the Structural Division, ASCE, April 1969, pp 661-675.

13. Whetstone, W. D., "Computer Analysis of Large Linear Frames," Journal of the Structural Division, ASCE, November 1969, pp 2401-2417.

14. IBM System/360 FORTRAN IV Language, File No. S360-25, Form C28-6515-7, IBM Corporation, Programming Publications, 1271 Avenue of the Americas, New York, N. Y. 10020.

15. Becker, E. B., Brisbane, J. J., and Schkade, Jr., A. F., "Investigation of Techniques of Three-Dimensional Finite Element Stress Analysis," Rohm and Haas Company, Redstone Research Laboratories, Huntsville, Alabama 35807, 1970.

16. Hanson, D. E., Three-Dimensional Elastostatic Problems Using Isoparametric Finite Elements (Mesh Generator), Master's Thesis, Naval Postgraduate School, Monterey, California 1971.

17. Leonidas, E., A General Purpose Three-Dimensional Stress Analysis Program, Master's Thesis, Naval Postgraduate School, Monterey, California, 1971.

18. Pfeifer, C. G., Evaluation of a Three-Dimensional Stress Analysis Program, Master's Thesis, Naval Postgraduate School, Monterey, California, 1972.

INITIAL DISTRIBUTION LIST

Library (Code 0212)
Naval Postgraduate School
Monterey, California 93940 — 2

Dean of Research (Code 023)
Naval Postgraduate School
Monterey, California 93940 — 2

Department of Mechanical Engineering
Naval Postgraduate School (Code 59)
Monterey, California 93940 — 1

Professor Gilles Cantin (Code 59Ci)
Naval Postgraduate School
Monterey, California 93940 — 5

Director
Defense Documentation Center
5010 Duke Street
Alexandria, Virginia 22314 — 20

DEPARTMENT OF DEFENSE

Defense Research and Engineering
Propulsion Technology Office
Pentagon 3D1065
Washington, D.C. 20301
ATTN: G. R. Makepeace — 1

AIR FORCE

Headquarters, U.S. Air Force (AFRDDG)
Pentagon Room 1D373
Washington D.C. 20301 — 1

Headquarters, U.S. Air Force
Pentagon, Room 1D373
Washington, D.C. 20330
ATTN: Lt. Col. Rufus D. Hutcheson — 1

Air Force Rocket Propulsion Laboratory (RPCL)
Edwards Air Force Base
Edwards, California 93523
ATTN: Forest S. Forbes — 1

Air Force Rocket Propulsion Laboratory (RPCL)
Edwards Air Force Base
Edwards, California 93523
ATTN: Norman J. Vander Hyde 1

Air Force Rocket Propulsion Laboratory (RPM)
Edwards Air Force Base
Edwards, California 93523
ATTN: Charles R. Cooke
Chief, Solid Rocket Division 1

Air Force Rocket Propulsion Laboratory (RPMCB)
Edwards Air Force Base
Edwards, California 93523
ATTN: Donald Saylak 1

Air Force Rocket Propulsion Laboratory (RPMCP)
Edwards Air Force Base
Edwards, California 93523
ATTN: Frank N. Kelley 1

Air Force Rocket Propulsion Laboratory (RPRE)
Edwards Air Force Base
Edwards, California 93523
ATT: Walter A. Detjen 1

Ogden Air Material Area
OOAMAA-OOMQOPOOMQQP
Hill Air Force Base, Utah 84401
ATTN: Merrill S. Budge, Bldg. 1917 1

ARMY

Commanding General (AMCRD-MT)
Army Material Command
T-7, Gravelly Point, Rm. 2705
Washington, D.C. 20315
ATTN: S. R. Matos 1

Commanding General (AMCRD-MT)
Army Material Command
T-7, Gravelly Point, Rm. 2705
Washington, D.C. 20315
ATTN: Harold E. S. Jersin 1

Commanding General (AMSMI-RKI)
Army Missile Command
Redstone Arsenal, Alabama 35808
ATTN: Dr. David C. Sayles 1

Commanding General (AMSMI-RKP)
Army Missile Command
Redstone Arsenal, Alabama 35808
ATTN: Thomas H. Duerr 1

Commanding General
Army Ballistic Research Laboratories
Aberdeen Proving Ground, Maryland 21005
ATTN: L. DeAngelis 1

Commanding General
Army Ballistic Research Laboratories
Aberdeen Proving Ground, Maryland 21005
ATTN: Alexander S. Elder 1

Commanding General
Picatinny Arsenal
Dover, New Jersey 07801
ATTN: Benjamin D. Lehman 1

Commanding General (SMUPA-DL)
Picatinny Arsenal
Dover, New Jersey 07801
ATTN: Irving Forsten 1

NAVY

Department of the Navy
Office of Naval Research (Code 439)
Washington, D.C. 20360
ATTN: John M. Crowley 1

Commander
Naval Air Systems Command (AIR-53664)
Room 2W21, W. Bldg.
Washington, D.C. 20360
ATTN: S. N. Block 1

Commander
Naval Air Systems Command (AIR-330)
Munitions Bldg., Rm. 3810
Washington, D.C. 20360
ATTN: Dr. O. H. Johnson 1

Commander
Naval Air Systems Command (AIR-330)
Munitions Bldg. Rm. 3810
Washington, D.C. 20360
ATTN: Irving Silver 1

Commander
Naval Air Systems Command (AIR-330D)
Munitions Bldg., Rm. 3810
Washington, D.C. 20360
ATTN: Robert H. Heitkotter 1

Commander
Naval Ordnance Systems Command (ORD-0331)
Munitions Bldg. Rm. 3612
Washington, D.C. 20360
ATTN: John W. Murrin 1

Commander
Naval Ordnance Station (Code RRCR-6)
Bldg. 888
Indian Head, Maryland 20640
ATTN: Garet Bornstein 1

Commander
Naval Ordnance Station
Indian Head, Maryland 20640
ATTN: L. A. Dickinson 1

Commander
Naval Ordnance Station
Indian Head, Maryland 20640
ATTN: Giorgio Tesi 1

Commander (Code 6058)
Naval Weapons Center
China Lake, California 93555
ATTN: Dr. Arnold Adicoff 1

Commander (Code 6058)
Naval Weapons Center
China Lake, California 93555
ATTN: Kenneth H. Bischel 1

Commander (Code 4505)
Naval Weapons Center
China Lake, California 93555
ATTN: Dr. Charles J. Thelen 1

Commander (Code 4584)
Naval Weapons Center
China Lake, California 93555
ATTN: Dean Couch 1

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

National Aeronautics and Space Administration Headquarters (Code RPS)
600 Independence Avenue, S.W., Rm. B625
Washington, D.C. 20546
ATTN: Robert W. Ziem 1

National Aeronautics and Space Administration Headquarters (Code RP)
600 Independence Avenue, S.W., Rm. B625
Washington, D.C. 20546
ATTN: Robert A. Wasel 1

National Aeronautics and Space Administration Headquarters (Code RPS)
600 Independence Avenue, S.W., Rm. B625
Washington, D.C. 20546
ATTN: Robert S. Levine 1

National Aeronautics and Space Administration
Langley Research Center
Langley Station
Hampton, Virginia 23365
ATTN: H. Jackson Hale 1

National Aeronautics and Space Administration
Langley Research Center
Langley Station
Hampton, Virginia 23365
ATTN: Robert L. Swain 1

National Aeronautics and Space Administration
Lewis Research Center
21000 Brookpark Road
Cleveland, Ohio 44135
ATTN: Irving A. Johnsen 1

CORPORATIONS

Aerojet-General Corporation
P.O. Box 15847
Sacramento, California 95813
ATTN: James H. Wiegand
Dept. 4720, Bldg. 0525 1

Aerojet-General Corporation
P.O. Box 15847
Sacramento, California 95813
ATTN: Kenneth W. Bills, Jr. 1

Aerospace Corporation
San Bernardino, California 93402
ATTN: John S. Wise 1

Applied Physics Laboratory
Chemical Propulsion Information Agency
8621 Georgia Avenue
Silver Spring, Maryland 20910
ATTN: Ronald D. Brown 1

Hercules, Inc.
Bacchus Works
Magna, Utah 84044
ATTN: James H. Thacher 1

Hercules, Inc.
Bacchus Works
Magna, Utah 84044
ATTN: S. C. Browning 1

Lockheed Propulsion Company
P.O. Box 111
Redlands, California 92374
ATTN: Walter D. Hart 1

Lockheed Propulsion Company
P.O. Box 111
Redlands, California 92374
ATTN: Harry Leeming 1

McDonnell-Douglas Corporation
Douglas Aircraft Company Division
3000 Ocean Park Boulevard
Santa Monica, California 90406
ATTN: R. V. Mellette, A2-260 1

Mathematical Sciences Northwest, Inc.
1107 Northeast 45th Street
Seattle, Washington 98105
ATTN: Dr. R.J.H. Bollard 1

North American Rockwell Corporation
Rocketdyne/Solid Rocket Division
McGregor, Texas 76657
ATTN: B. L. Black 1

North American Rockwell Corporation
Rocketdyne/Solid Rocket Division
P.O. Box 548
McGregor, Texas 76657
ATTN: J. L. Fulbright 1

North American Rockwell Corporation
Rocketdyne/Solid Rocket Division
P.O. Box 548
McGregor, Texas 76657
ATTN: B. C. Harbert 1

Rohm and Haas Company
Huntsville, Alabama 35808
ATTN: Charles H. Parr
Rohm and Haas Company
Huntsville, Alabama 35808
ATTN: Max Sigmon 1

Susquehanna Corporation
Atlantic Research Group
Shirley Highway at Edsall Road
Alexandria, Virginia 22314
ATTN: Courtland N. Robinson 1

Thiokol Chemical Corporation
Wasatch Division
Brigham City, Utah 84302
ATTN: S. John Bennett 1

Thiokol Chemical Corporation
Huntsville Plant
Huntsville, Alabama 35800
ATTN: R. B. Kruse 1

TRW Systems, Inc.
One Space Park
Redondo Beach, California 90278
ATTN: J. L. Myers, Rl/2170 1

United Technology Center
P.O. Box 358
Sunnyvale, California 94088
ATTN: Richard A. Jankowski 1

United Technology Center
P.O. Box 358
Sunnyvale, California 94088
ATTN: Eugene C. Francis 1

UNIVERSITIES

California Institute of Technology
1201 E. California Boulevard
Pasadena, California 91109
ATTN: Security Officer
(For Prof. W. G. Knauss) 1

California Institute of Technology
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91103
ATTN: Robert F. Landel 1

Stanford Research Institute
Propulsion Sciences Division
Menlo Park, California 94025
ATTN: Norman Fishman 1

Texas A&M University
209-215 Engineering Building
College Station, Texas 77843
ATTN: Assistant Director A.D. Rychlik
(for: S.W. Beckwith; S. C. Britton;
R.A. Schapery; L. D. Webb) 1

University of Utah
Salt Lake City, Utah 84112
ATTN: Security Officer
(for: Prof. J.E. Fitzgerald;
Prof. M. L. Williams) 1

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California	UNCLASSIFIED
	2b. GROUP

REPORT TITLE

Three-Dimensional Finite Element Studies. Part One: Service Routines

DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Technical Report, 1965

AUTHOR(S) (First name, middle initial, last name)

Cantin, Gilles

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS

a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	NPS-59CI72121A
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

0. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

1. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Weapons Center China Lake, California

3. ABSTRACT

In this report, service routines are developed for linear, quadratic and cubic finite elements for two and three dimensional regions. Also, an equation solver of very large capacity is developed for systems of linear equations where the matrix of coefficients is symmetrical, positive definite and tightly banded along the main diagonal.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

UNCLASSIFIED

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Isoparametric						
Finite Elements						
Computer Code						
Linear Equation Solver						